Sonderheft Nr. 08
Preis DM 14.80
öS 114.–, sfr. 14.80
Funkschau

ZX 81 Kochbuch

Für Anfänger & Fortgeschrittene

Tips & Tricks

Bauanleitungen

Arbeitshilfen

Wenn Sie die Kommunikations- und Unterhaltungselektronik immer im Auge behalten wollen:

Die Funkschau informiert in Fachbeiträgen alle 14 Tage umfassend zu den Themen Kommunikations- und Computertechnik, Audio und Video.

Technische Beiträge

Beschreibung von Geräten, Systemen und ihren Funktionen – einschließlich technischer Details, wenn sie für den Anwender von Bedeutung sind.

Erfahrungsberichte

Geräte der elektronischen Kommunikationstechnik und der Konsumelektronik im praktischen Gebrauch. Die FUNKSCHAU beschreibt den Nutzen für den Anwender.

Produktneuheiten

Das Schaufenster des Elektronik-Marktes. Neue Produkte werden mit den wichtigsten technischen Daten, Anwendungsbeispielen und Bezugsquellenangaben präsentiert.

Service-Beiträge

Arbeitshilfe für Service-Techniker aller Elektronikbereiche: Arbeitsblätter mit Grundschaltungen der Elektronik, Schaltungsdetails, Schaltungsapplikationen, Meßtechnik, Fehlerquellensuche...

Bauanleitungen

Für den anspruchsvollen Hobby-Elektroniker zum sicheren Nachbau. Beispiele: Meßgeräte, Videomischpulte, Heizungsregelung per Computer, Alarmanlagen, Musiksynthesizer...

Reports

Aktuelle Elektronik-Themen aus unterschiedlichen Blickwinkeln betrachtet. Runduminformationen, die neben der Technik auch gesellschaftspolitische Aspekte neuer Entwicklungen beleuchten.

Nachrichten

Was für den Leser zur Beurteilung der Technik des Elektronik-Marktes wichtig ist, wird übersichtlich aufbereitet und mit Stellungnahmen ergänzt.

Wenn Sie sich vom Nutzen der FUNKSCHAU für Ihre Arbeit überzeugen wollen, schicken wir Ihnen gerne im Rahmen unseres Kennenlern-Angebots die neueste Ausgabe kostenlos und unverbindlich ins Haus.
Die vorbereitete Karte finden Sie an der Umschlagklappe.

Funkschau
Zeitschrift für Unterhaltungselektronik und Kommunikationstechnik

Liebe Leser,

nach wie vor gilt die Binsenwahrheit, daß ein Computer nur so gut ist, wie die für ihn verfügbare Software. Passiven Computer-Hobbyisten wird deshalb eine Unmenge an Programmen angeboten. Jedoch, die Sache hat einen Haken: Gilt es per Computer eine individuelle Aufgabe zu lösen, dann bleibt vom vermeintlich allumfassenden Programmangebot oft nichts mehr übrig. Gar vollends im Regen stehengelassen wird man, wenn sich eine Aufgabe nicht mehr allein mit Software lösen läßt und Hardware ins Spiel kommt. Oder kennen Sie vielleicht einen Computer, der z. B. mit tollen Lichteffekten Stimmung in eine lahme Party bringt? Nein, ein solches Modell gibt es leider nirgends. Dennoch brauchen Sie die Flinte nicht gleich ins Korn werfen: Schlagen Sie lieber das ZX-81-Kochbuch auf.

Wir haben hier 52 „Rezepte" für Sie zusammengestellt. Davon sind 20 brandneu und die übrigen eine Auslese der ZX-81-Beiträge, die in den letzten Monaten von der Zeitschrift FUNKSCHAU veröffentlicht wurden. Damit haben Sie ein „Informationspaket" darüber vorliegen, wie Sie den ZX 81 noch besser durchschauen und bisher unlösbare Aufgaben bewältigen.

Freilich, wenn Sie die Bauanleitungen nutzen möchten, um dem Computer das Tor zum Umfeld aufzustoßen, dann erwarten wir, daß Sie schon mit einem Lötkolben umgehen können. Und sollte Ihnen die Z-80-Maschinensprache noch kalte Schauer über den Rücken jagen, dann führen Sie sich einmal unser Sonderheft „Klartext für den ZX 81" zu Gemüte. Wenngleich das für die Nutzung der vorliegenden „Kochrezepte" nicht unbedingt erforderlich ist, so sind Kenntnisse in Maschinensprache zum Verständnis einiger Beiträge Voraussetzung.

Damit sind wir beim Stichwort „Verständnis", denn genau das wollen Ihnen unsere Beiträge für den ZX 81 vermitteln: Je mehr Sie über den ZX 81 wissen, desto besser können Sie mit ihm Ihre individuellen Ideen verwirklichen – ohne zwangsläufig auf fremde Hilfe angewiesen zu sein. Das erst verleiht dem Computer-Hobby das Tüpfelchen aufs I.

Jede Menge Tips & Tricks, Schaltungsvorschläge, Arbeitshilfen und reizvolle Blicke hinter die Kulissen des ZX 81 – das ist unser Rezept, mit dem wir bei Ihnen Verständnis für den Computer wecken möchten, ob Sie jetzt Anfänger sind oder schon Fortgeschrittener. Und wenn Ihnen das, was Sie sehen und erfahren, gefällt, dann schauen Sie doch einmal in die aktuellen Hefte der FUNKSCHAU. Hier betreuen wir den ZX 81 weiter mit attraktiven Beiträgen. Bestimmt finden auch Sie dann Geschmack z. B. an einem Joystick-Interface, einem DCF-77-Decoder, einer Serie über den ZX 81 als Meßinstrument oder an Methoden, wie sich Programme dem fremden Zugriff entziehen lassen. Deshalb sagen wir nicht Ade, sondern auf Wiedersehen.

Ihre FUNKSCHAU-Redaktion

Inhaltsverzeichnis

Fachaufsätze

Software

Nutzen von ROM-Routinen:
Kammerjäger 3
Z-80-Disassembler:
Übersetzer für Maschinencode 4
Ausgabe von Morsezeichen:
Vom Klartext zum Maschinencode 7
Erfahrungsbericht: Tempo, Tempo
Das Schnelladeprogramm GM Save 39
Entschlüsseln von Morsezeichen 51
ZX-81-intern:
Das ROM hilft gerne mit 53
Arbeitshilfe:
Raumgestalter für Maschinencode 55
Programmieren in Maschinensprache:
Z-80-Training mit dem ZX 81 57
Arbeitshilfe:
Schieber für Maschinencode 59
Verknüpfen von Basic-Programmen:
Brückenbauer 63
Renumber-Routine:
Zeilen-Verwalter 65
Grundlagen:
Eingabe von Maschinencodes 72
ZX-81-Betriebssystem:
RAMTOP-Story 74

Hardware

ZX 81 à la carte (3):
Software voller Klang 12
ZX 81 à la carte (4):
Schrittmacher (Auto-Repeat) 16
ZX 81 à la carte (5):
Daten-Drehscheibe (PIO-Port), Teil 1 18
ZX 81 à la carte (6):
Daten-Drehscheibe, Teil 2 22
ZX 81 à la carte (7):
Daten-Drehscheibe, Teil 3 23
Nachtrag zur Daten-Drehscheibe:
Entfesselte PIO 26
ZX 81 à la carte (8):
Hardware voller Klang, Teil 1 28
ZX 81 à la carte (9):
Hardware voller Klang, Teil 2 30
ZX 81 à la carte (10):
Telefoncomputer 33
ZX 81 à la carte (11):
Nachwuchs für den Ausgabe-Port 36
ZX-81-Schaltplan 40
Erfahrungsbericht: Grafikmodul
Bild-Meister 42
Computer lernen sprechen:
Sprachschatz 43
Vollständige ROM-Decodierung:
Wiederbelebung 48
Messen mit dem ZX 81:
Frequenzmesser 61
Programmierbare Lichtorgel:
Licht im Takt 66
8-Bit-Datenausgabe:
Spar-Interface 70

Kurzbeiträge

Software

Zwei Zeilen mehr 10
Zahlenkolonnen im Griff 10
Schiebung im Speicher 11
Wiederfinden von Programmteilen 17
Ladekontrolle durch Prüfsumme 17
Simulieren der Fixkomma-Darstellung 25
Zeit sparen bei SAVE und LOAD 27
Stopper für LIST 54
Ersatz für Pause-Befehl 62
Laufschrift 69
Listen am laufenden Band 73
Soviele Bytes sind frei 80

Hardware

Verbesserungsvorschlag – Auto-Repeat 6
Verriegelung für SHIFT 6
Sensor-SHIFT 11
Tastatur-Pieps 34
Lange Leitung 54
Alte Batterien nutzen 58
Störungsfreies Fernsehbild 58
Wissenswertes zum RAM-Modul 69
Tempo durch Nachbrenner 71

Impressum:

1986 · 2., verbesserte Auflage
Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2. Bearbeitet von der Redaktion der Zeitschrift FUNKSCHAU.
Für den Text verantwortlich: Ing. (grad.) Stephan Schall. Titelbild: Studio Hager.

© Sämtliche Rechte – besonders das Übersetzungsrecht – an Text und Bildern vorbehalten. Fotomechanische Vervielfältigung nur mit Genehmigung des Verlages. Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.

ISSN 0172-2778

Druck: Franzis-Druck GmbH, München. Printed in Germany. Imprimé en Allemagne
ZV-Artikel Nr. 08011 · F/ZV/886/1014a/2'

Nutzen von ROM-Routinen:

Kammerjäger

Sollen im Verlauf eines Maschinenprogramms Zeichen oder Texte am Bildschirm ausgegeben werden ist ein Rücksprung ins Basic unnötig, sofern man sich die PRINT-Routine im ROM zunutze macht.

Die PRINT-Routine im ROM des ZX 81 wird mit *rst 10h* (entspricht *rst 16*), also mit einem speziellen Z-80-Befehl aufgerufen. Dieser Befehl schreibt das im Akku stehende einstellige Zeichen an den linken oberen Bildrand, worauf die PRINT-Position automatisch um eine Spalte weitergerückt wird (*Bild 1, oben*).

Bei längeren Texten ist es speicherplatzschonender, wenn die PRINT-Routine in einer Schleife aufgerufen wird, wobei die auszugebenden Zeichen am Anfang in einer Tabelle eingetragen werden. Gibt man z. B. in eine REM-Zeile mit Startadresse 16514 folgende Hex-Codes ein: 2B 3A 33 30 38 28 2D 26 3A 00 und anschließend (ab Adresse 16524) das in Bild 1 (*Mitte*) gezeigte Programm, dann wird nach RAND USR 16524 der Text FUNKSCHAU geschrieben.

Das Programm setzt zuerst Register b als Schleifenzähler für zehn Zeichen, lädt das hl-Registerpaar mit der Adresse des ersten Zeichens und lädt dieses Zeichen in den Akku. Dann wird die PRINT-Routine aufgerufen und das hl-Registerpaar um 1 erhöht. Solange das b-Register nicht auf 0 steht, wiederholt sich der ganze Vorgang.

Eine positionierte Ausgabe ist mit Hilfe des bc-Registerpaares und einer ROM-Routine bei Adresse 2293 möglich. So schreibt das Programm in Bild 1 (*unten*) den Text „ZX-81" in die Bildmitte. Allerdings benutzt die ROM-Routine ebenfalls das hl- und bc-Registerpaar, so daß die gewählten Werte nach der Rückkehr aus der Routinen verloren sind. Werden die Werte weiterhin benötigt, müssen beide Registerpaarinhalte vor dem Aufruf der Routine mit *push* auf den Stapel gelegt und nach Ablauf der Routine mit pop wieder abgehoben werden. Außerdem kann es mit RAM-Zusatzspeichern nötig sein, mit POKE 16389,68 einen „Spar-Bildschirm" (siehe Seite 74) aufzubauen, sollte es bei der Ausgabe der Buchstaben bzw. Texte zu „Geistererscheinungen" kommen.

```
4082 LD A,063          3E3F
4084 RST 10            D7
4085 RET               C9


408C LD B,010          060A
408E LD HL,16514       21B240
4091 LD A,(HL)         7E
4092 RST 10            D7
4093 INC HL            23
4094 DJNZ $4091        10FB
4096 RET               C9


4082 LD B,011          060B
4084 LD C,013          0E0D
4086 CALL 02293        CDF508
4089 LD A,063          3E3F
408B RST 10            D7
408C LD A,061          3E3D
408E RST 10            D7
408F LD A,022          3E16
4091 RST 10            D7
4092 LD A,036          3E24
4094 RST 10            D7
4095 LD A,029          3E1D
4097 RST 10            D7
4098 RET               C9
```

① **Textausgabe:** Ausgabe eines einzelnen Zeichens (oben), eines zuvor in zehn Speicherzellen abgelegten Textes (Mitte) und positionierte Ausgabe (unten).

Das Spielprogramm „Kammerjäger" nutzt diese Art der Zeichenausgabe. Seine Hex-Codes aus *Bild 2* sind in eine 295 Zeichen lange REM-Zeile unterzubringen. Anschließend ist das Eingabeprogramm (siehe Seite 72) mit den folgenden Zeilen zu überschreiben:

```
20 RAND USR 16519
30 INPUT N$
40 CLS
50 RUN
```

Durch das Überschreiben wird ein mögliches „Endloslisten" verhindert. Nach dem Start mit RUN huscht am linken Bildrand ein Kammerjäger umher, der sich gesteuert mit den links und rechts außen angeordneten Tasten Mäusen erwehren muß. Ziel ist es, die Mäuse an den rechten Bildrand zu treiben. Wem das nicht gelingen will, kann die Speicherzellen 16516 bis 16518 mit 0 laden – das vereinfacht die Aufgabe (kein SCROLL).

In Adresse 16514/15 wird die Position einer Maus festgelegt, dann folgt bis 16518 die SCROLL-Routine. Von 16519 bis 16537 wird die Maus, von 16538 bis 16596 der Jäger „ausgegeben". Dann, bis 16628: Bewegung des Jägers. Anschließend bis 16636 erfolgt die Tastaturabfrage. Von 16637 bis 16660 erfolgt die Pfeil-Ausgabe und bis 16677 die Trefferabfrage. Dann wird bis 16734 die nächste Pfeil- und Mausposition berechnet. Die Kommentar-Ausgabe ab 16735 schließt das Programm ab.

Paul Webranitz/-ll

```
ADR.    HEX-DUMP
16514   12 03 CD 0E 0C ED 4B B2 40 CD F5 08 3E 12 D7 3E B3 D7 3E
16533   1B D7 3E 00 D7 00 00 00 00 00 00 00 00 00 00 06 07 0E
16552   00 C5 CD F5 08 3E 00 D7 C1 04 C5 CD F5 08 3E 34 D7 C1 04
16571   C5 CD F5 08 3E 07 D7 C1 04 C5 CD F5 08 3E B2 D7 C1 04 C5
16590   CD F5 08 3E 00 D7 C1 3A A6 40 FE 01 28 0B FE 0E 28 0B 00
16609   3D 32 A6 40 18 0E 3E 3C 18 02 3E 3D 32 E1 40 3A A6 40 18
16628   EC AF DB 00 CB 47 C2 A5 40 3A A6 40 32 B3 40 00 47 0E 01
16647   C5 CD F5 08 3E 00 D7 3E 16 D7 3E 12 D7 C1 00 00 00 00 00
16666   00 00 00 00 00 00 00 2A 0E 40 3E B3 BE CA 48 41 21 06 41
16685   3E 1C BE 28 04 34 C3 FD 40 3E 01 32 06 41 21 B2 40 35 3E
16704   02 BE CA 7E 41 C3 B4 40 3E 00 D7 21 B2 40 34 34 00 00 00
16723   00 00 00 00 00 00 00 3E 1A BE C2 FD 40 18 21 37 2A 2C 2A
16742   2E 38 00 37 2A 2C 2A 26 2F 37 2A 2C 2A 2E 38 00 2A 38 3A
16761   2A 26 32 00 00 21 6F 41 18 03 21 62 41 06 0D 0E 03 C5 E5
16780   CD F5 08 E1 7E E5 D7 E1 C1 23 10 F2 3E 12 32 B2 40 3E 03
16799   32 B3 40 C9 C9
```

② **Kammerjäger-Programm:** Ein Eingabeprogramm muß die Hex-Codes in einem REM-Kommentar unterbringen. Die nop's erleichtern Programmänderungen

Z-80-Disassembler:

Übersetzer für Maschinencode

Maschinencode ist allein für den Mikroprozessor des ZX 81 eine leichtverständliche Sache. Programmierer kommen mit den mnemonischen Kürzeln viel besser zurecht. Die Übersetzung der Hex-Codes übernimmt ein Z-80-Disassembler in Form eines Maschinencodeprogramms.

Wer schon einmal versucht hat, sich durch ein Maschinencodeprogramm durchzuarbeiten, weiß, daß es sehr mühsam ist, den Hex-Code mittels einer Befehlsliste von Hand in die mnemonischen Kürzel zu übersetzen. Außerdem unterlaufen einem dabei leicht Fehler. Es handelt sich jedoch um eine rein „mechanische" Arbeit, die ebensogut durch ein Computerprogramm erledigt werden könnte – und diese Aufgabe übernimmt ein Disassembler. Er übersetzt beispielsweise die hexadezimale Ziffernfolge 21 48 03 in die verständlichere Form ld hl,S0348.

Oberhalb von RAMTOP ist der Disassembler sicher

Der hier beschriebene Z-80-Disassembler ist für den ZX 81 mit mindestens 16 KByte RAM geeignet. Er ist selbst in Maschinensprache abgefaßt und benötigt daher nur relativ wenig Speicherplatz (1370 Byte). Der Disassembler arbeitet schnell und läßt sich vor allem gut vor dem Löschen geschützt im Speicher unterbringen, so daß er auch ein nachträglich von Kassette geladenes Programm disassemblieren kann. Das Ausdrucken eines so erstellten Assembler-Listings ist allerdings nicht möglich.

Die Eingabe des Disassemblers sollte genau nach folgendem Schema ablaufen. Zunächst tippt man ein: POKE 16389,99 und anschließend NEW. Dadurch wird Speicherplatz im oberen Bereich des RAMs reserviert, der zur späteren Eingabe des Maschinencodes benötigt wird.

Nun tippt man das in *Bild 1* gezeigte Basic-Listing ein, das ab Zeile 120 ein Eingabeprogramm für Hex-Codes enthält. Hinter der REM-Anweisung in Zeile 1 müssen mindestens 64 beliebige Ziffern oder Buchstaben eingegeben werden, wiederum, um Speicherplatz für Maschinencode zu schaffen. Dieser Maschinencode – es handelt sich um ein Ladeprogramm, dessen Wirkung später erläutert wird – ist als nächstes einzugeben (Start mit RUN 120 und Eintippen der in *Bild 2* gezeigten Hex-Codes).

Anschließend wird nach erneutem Programmstart mit RUN 130 der Hex-Code des eigentlichen Disassemblers auf die gleiche Weise eingegeben (*Bild 3*); er findet Platz in dem anfangs reservierten Speicherplatz. Dieser Hex-Code stellt eine lange Tabelle, die Informationen über den Aufbau der mnemonischen Kürzel enthält, sowie Maschinenroutinen dar.

Schließlich wird durch RUN 70 der Hex-Code des Disassemblers in die Variable M$ übertragen. Von jetzt an dürfen die Befehle RUN und CLEAR nicht mehr benutzt werden, da sonst diese Variable gelöscht würde!

Alle Programmzeilen ab Zeile 70 darf man nun wieder löschen; sie haben keine Bedeutung mehr. Jetzt kann und sollte der Disassembler auf Kassette gespeichert werden, was durch GOTO 50 möglich ist. Der Hex-Code, der sich ja inzwischen in M$ befindet, gelangt mit aufs Band.

```
   1 REM 64 BELIEBIGE ZEICHEN
  10 PRINT TAB VAL "10";"DISASSE
MBLER",,,"STARTEN DURCH: RUN USR
 32268",,,"PRUEFSUMME = ";USR VA
L "16514"
  20 PRINT VAL "PEEK 16434+256*P
EEK 16435"
  30 IF VAL "PEEK 3883<>253" THE
N POKE VAL "32326",VAL "40"
  40 STOP
  50 SAVE "DIS"
  60 GOTO 1
  70 DIM M$(1370)
  80 FOR I=1 TO 1370
  90 LET M$(I)=CHR$ PEEK (31389+
I)
 100 NEXT I
 110 STOP
 120 FOR A=16514 TO 16577
 125 GOSUB 200
 127 NEXT A
 128 STOP
 130 FOR A=31390 TO 32759
 140 GOSUB 200
 150 NEXT A
 160 STOP
 199 PRINT AT 10,10;"FEHLEINGABE
";AT 11,6;"EINGABE WIEDERHOLEN"
 200 INPUT A$
 205 CLS
 208 IF A$="S" THEN STOP
 210 IF LEN A$<>2 THEN GOTO 199
 220 FOR I=0 TO LEN A$ STEP 2
 230 POKE A+INT (I/2),16*(CODE A
$(I+1)-28)+CODE A$(I+2)-28
 240 RETURN
```

① **Hilfsprogramm:** Dieses Programm ermöglicht das Speichern und Wiedereinlesen des Disassemblers oberhalb von RAMTOP. Außerdem ist ein Eingabeprogramm für die Hex-Codes des Transfer- und Disassembler-Programms darin enthalten

So wird mit dem Disassembler gearbeitet

Der Disassembler startet nach dem Laden selbständig, sofern er durch GOTO 50 gespeichert wurde. Es wird dann das Transfer-Programm in Zeile 1 mit Startadresse 16514 aufgerufen, das den Inhalt der Systemvariablen RAMTOP auf 31390 heruntersetzt und das Disassembler-Programm aus M$ in den Speicherbereich ab Adresse 31390 überträgt (ein Programmstart mit RUN würde M$ löschen!). Gleichzeitig wird zur Kontrolle eine Prüfsumme gebildet und anschließend ausgegeben. Sie muß 54028 betragen; anderenfalls wurde der Hex-Code fehlerhaft geladen oder eingegeben.

Der Disassembler steht nun zur Verfügung und ist vor Löschung, z. B. durch NEW, weitgehend geschützt. Das Basic-Programm kann nun gelöscht oder ein anderes Programm geladen werden. Durch RUN USR 32268 wird anschließend der Disassembler aufgerufen.

Zuerst erfragt dieser die Adresse, ab der disassembliert werden soll. Die Adresse ist vierstellig hexadezimal einzutasten. Durch RUBOUT kann eine fehlerhafte Eingabe gelöscht werden, und durch BREAK während der Adreßeingabe wird wieder ins Basic zurückgesprungen.

Wird während der Adreßeingabe jedoch die Taste H gedrückt (erscheint nicht auf dem Bildschirm), so wird anschließend nicht disassembliert, sondern ein reines Hex-Listing erzeugt. Durch ein M bei der nächsten Adreßeingabe wird wieder auf mnemonisches Disassemblieren umgeschaltet.

Das Disassemblieren erfolgt zeilenweise; jeder Druck auf eine beliebige Taste bewirkt, daß die nächste Zeile abgebildet wird. Durch BREAK verzweigt das Programm wieder zur Adreßeingabe. Eine Disassemblerzeile besteht aus Befehlsadresse, hexadezimalem und mnemonischem Befehlsausdruck.

```
ADR.    HEX-DUMP

16514   21 9E 7A 22 04 40 F9 01
16522   00 3E C5 21 76 06 E5 ED
16530   73 02 40 41 2A 10 40 7E
16538   FE D2 28 02 CF 01 23 23
16546   23 23 5E 23 56 D5 ED 5B
16554   04 40 23 7E 12 B1 4F 88
16562   91 47 13 E3 28 7C B5 E3
16570   20 F0 ED 43 32 40 CF FF
```

② **Transferprogramm:** Nach dem Start des Hilfsprogramms mit RUN 120 sind diese Hex-Codes einzeln einzugeben (immer zweistellige Eingabe und mit NEWLINE abschließen)

Alle Zahlenangaben innerhalb eines Befehlskürzels erfolgen hexadezimal und werden durch ein $-Zeichen (signalisiert Hex-Zahl) eingeleitet. Die Sprungziele relativer Sprungbefehle werden als Absolutadresse angezeigt; ein umständliches Umrechnen relativer Adressen ist also nicht nötig.

```
31390   2F 37 00 02 1A 57 33 34 75 2A 3D 00 26 2B 1A 26 2B 4B 29
31409   2F 33 3F 57 2F 37 00 D7 44 36 03 1A 7C 26 29 29 00 0A 1A
31428   C3 42 36 10 27 2B 11 1A 66 36 26 1A 10 27 2B 51 36 10 29
31447   2A 11 1A 66 36 26 1A 10 29 2A 51 36 10 3C 11 1A 4A 36 0A
31466   1A 10 3C 51 36 10 3C 11 1A 66 36 26 1A 10 3C D1 44 2E 33
31485   2B 00 43 29 2A 2B 00 C3 2E 33 2B 00 C4 29 2A 2B 00 C4 36
31504   04 1A FB 42 37 31 2B 66 37 37 2B 66 37 31 66 37 37 66 29
31523   26 66 2B 35 71 3B 2B 6B 2B 2B EB 36 04 1A 45 BA 06 45 8A
31542   37 2A 39 00 C1 45 35 34 35 00 47 37 2A 79 2A 3D 7D 2F 35
31561   00 10 0A 51 36 3B 35 1A CA 2F 35 00 01 1A FC 42 2F 35 00
31580   7C 56 34 3A 39 00 10 3B 11 1A 66 2E 33 00 26 1A 10 3B 51
31599   2A 3D 00 10 3B 35 11 1A 4A 2A 3D 00 29 2A 1A 2D 71 29 6E
31618   2A EE 2B 26 31 31 01 1A FC 44 35 3A 3B 2D 47 2B 26 31 31
31637   FC 06 FB 46 1C 5C 1C 64 1D 5C 1D 64 1E 5C 1E 64 1F 5C 1F
31656   E4 42 37 31 2B 00 45 37 37 2B 00 45 37 31 00 45 37 37 00
31675   45 3B 31 26 00 45 3B 37 26 00 45 56 3B 37 31 00 45 BA 27
31694   2E 39 00 08 1A 45 BA 37 2A 3B 00 08 1A 45 8A 3B 2A 39 00
31713   0B 1A 45 BA 56 BA 2E 33 00 04 1A 10 2B D1 34 3A 39 00 10
31732   2B 11 1A C4 44 38 27 2B 00 2D 31 1A 43 26 29 2B 00 2D 31
31751   1A C3 44 36 10 3C 11 1A 43 36 03 1A 10 3C D1 33 2A EC 44
31770   37 2A 39 73 37 2A 39 EE 42 2E 32 5C 56 2E 32 5D 2E 32 5E
31789   56 56 56 D6 42 36 2E 1A 66 36 37 1A 66 36 26 1A 6E 36 26
31808   1A 77 37 37 69 37 76 56 D6 43 31 69 2B 75 2E 73 34 79 56
31827   56 56 56 0C 56 56 56 56 6E 69 2E 77 29 77 8A 56 8A 33 7F
31846   7F 33 68 68 35 74 35 6A 75 72 27 6B 29 6A 4A 38 75 67 68
31865   69 6A 6D 71 49 66 26 29 29 00 26 5A 26 29 2B 00 26 5A 38
31884   3A 27 40 38 27 2B 00 26 5A 26 33 29 40 3D 34 37 40 34 37
31903   40 2B 35 40 27 68 29 6A 4A 26 6B 2D 71 2E 7D 2E 7E 10 2D
31922   31 51 10 2E 3D 15 3B 51 10 2E 3E 15 3B 51 E5 C5 CD D2 7C
31941   3C 20 FA CD D2 7C 3C 2B FA C1 E1 3D C9 CD BB 02 7D 2C C8
31960   44 4F CD BD 07 7E C9 E3 CD F2 7C E3 C9 C5 47 A7 28 07 CB
31979   76 23 2B FB 10 F9 C1 7E E6 3F CD 30 7D CB 76 23 28 F5 C9
31998   2A 32 40 7E 23 22 32 40 C3 13 7D E1 7E 23 E5 00 3D F8 D7
32017   1B F6 F5 F5 E6 F0 1F 1F 1F 1F C6 1C 47 D7 F1 E6 0F C6 1C
32036   4F D7 F1 C9 78 CD 30 7D 79 C3 30 7D FE 36 20 07 CD DF 7C
32055   31 29 40 C9 E5 C5 FE 3B 20 1B CD DF 7C 4D 2A 7B 40 25 20
32074   0B 7D ED 4B 3F 40 A7 20 03 CD FE 7C CD 2B 7D C1 E1 C9 FE
32093   17 20 24 CD DF 7C 4D CD FE 7C 4F 06 00 B7 30 01 05 09 E5
32112   01 14 15 CD F5 08 E1 7C CD 13 7D CD 2B 7D 7D CD 13 7D 1B
32131   D2 FE 3C 20 11 CD FE 7C C5 CD FE 7C CD DF 7C 4D CD 2B 7D
32150   C1 18 BD C1 21 64 7C FE 02 20 07 3A 3D 40 D6 04 1B 07 FE
32169   01 20 08 3A 3D 40 CD E5 7C E1 C9 FE 03 20 0A 21 70 7C 3A
32188   3D 40 CB 3F 1B ED 21 77 7C FE 04 28 E3 FE 05 20 05 3A 3C
32207   40 1B DD 21 7F 7C FE 06 2B D3 21 A3 7C FE 07 28 DB FE 0A
32226   20 08 21 AA 7C 3A 7B 40 1B C3 21 B0 7C FE 09 2B F4 FE 08
32245   20 05 3A 3D 40 C6 1C F5 21 5D 40 7E 23 3C 20 FB 3D 77 2B
32264   F1 77 E1 C9 FD 36 22 00 01 00 16 CD F5 08 CD 09 7D 2A 2F
32283   39 27 39 39 2B 33 2B 32 2B 38 77 2B 3F 01 33 2F 29 2E 27
32302   2B 32 01 39 29 2E 38 27 33 33 1B 01 31 2F 2B 32 00 21 32
32321   02 22 21 40 CD 2B 0F CD 0E 0C 01 00 15 CD F5 08 CD 09 7D
32340   27 2A 3B 1C 0F 01 01 01 01 01 00 01 06 15 CD F5 08 21 5D
32359   40 06 04 CD C0 7C 20 01 CF FE 2D 28 04 FE 32 20 05 32 21
32378   40 18 ED FE 77 2B CA FE 1C 38 E5 FE 2C 30 E1 77 23 D7 10
32397   DC 11 5D 40 06 02 1A D6 1C 17 17 17 17 65 6F 13 1A D6 1C
32416   B5 6F 13 10 EE 22 32 40 CD 0E 0C 2A 32 40 7C CD 13 7D 7D
32435   CD 13 7D AF D7 3A 21 40 FE 2D 20 0E AF D7 16 08 AF D7 CD
32454   FE 7C 15 20 FB C3 EF 7F FD 36 5D FF 0E 00 59 C5 CD FE 7C
32473   C1 57 E6 07 6F 7A E6 38 1F 1F 1F 67 22 3C 40 7A E6 C0 07
32492   07 32 3E 40 7B A7 20 44 7A FE 76 20 09 CD DF 7C 2D 26 31
32511   79 1B 2B FE CB 20 0F 1C 79 A7 28 CA C5 CD FE 7C ED 43 3F
32530   40 18 C1 FE ED 20 04 1E 02 18 B8 FE DD 2B 04 FE FD 20 13
32549   0D 0C 2B 07 CD DF 7C 56 C3 CE 7F 0C FE DD 2B 01 0C 18 E3
32568   63 69 22 7B 40 7B 07 07 47 3A 3E 40 80 17 17 17 47 3A 3C
32587   40 57 FD 5E 3D 80 47 21 9E 7A 20 0B 3A 3D 40 D6 04 30 1F
32606   C6 05 18 42 7E 23 CB 7F 2B FA FE 8A 20 13 78 D6 07 47 3B
32625   02 20 0B 2B 2B CB 7E 28 FB 23 1B 04 18 4C 10 E1 7E CB 77
32644   2B F7 23 FE 41 20 06 CD DF 7C 76 18 12 FE 42 2B 0E FE 43
32663   20 10 7A CD E5 7C 7E 23 FE 0C 20 FA 7B CD E5 7C 1B 25 FE
32682   44 20 05 7B E6 01 18 F2 FE 45 20 0B 7B E6 01 2B E9 43 CB
32701   38 80 18 E3 CD DF 7C 37 38 39 00 4D 18 D8 CD F2 7C 01 0F
32720   15 CD F5 08 21 5D 40 06 04 7E 34 2B 12 23 A7 28 03 D7 10
32739   F5 E5 01 14 15 CD F5 08 E1 44 18 EA CD C0 7C C2 A8 7E C3
32758   48 7E
```

③ **Disassembler:** Zur Eingabe der Hex-Codes des Disassemblers muß das Hilfsprogramm mit RUN 130 gestartet werden

Vom Hex-Code zum Mnemonik-Kürzel

Ein erheblicher Teil des benötigten Speicherplatzes, nämlich 546 Byte, wird von Tabellen eingenommen. Die erste Tabelle (mit 454 Byte die längste) steuert weitgehend das Disassemblieren. Der erste Teil der Tabelle wird auf dem Bildschirm sichtbar, wenn nach dem Laden des Programms PRINT M$ eingegeben wird. Man findet bruchstückhaft einzelne Mnemonics. Die vielen Fragezeichen, Sonderzeichen und ZX-Schlüsselworte stehen für Sonderfunktionen, die teilweise durch weitere Tabellen definiert werden.

Wie der Disassembler arbeitet, soll an einem Beispiel erläutert werden. Angenommen, der Disassembler stößt auf ein zu disassemblierendes Byte mit dem hexadezimalen Inhalt 79. Zunächst wird das Byte oktal (das heißt im Achtersystem, es gehören also drei Bits zu einer Oktalziffer) interpretiert; das ergibt den Wert 171. Die drei Oktalziffern werden in dieser Reihenfolge den drei „Variablen" F, G und H zugewiesen (F = 1, G = 7, H = 1).

Die Variablen sind durch sonst ungenutzte Speicherplätze im Bereich der Systemvariablen dargestellt. In Abhängigkeit von F (F kann Werte von 0 bis 3 annehmen) wird nun auf einen bestimmten Teil der großen Tabelle zugegriffen und der Inhalt bis zu einer Endmarke gelesen. Sie ist dadurch gekennzeichnet, daß Bit 6 in dem letzten zu lesenden Byte auf 1 gesetzt ist. Im vorliegenden Fall wird gelesen: *ld r(G), r(H)*.

Allerdings sind die Bestandteile dieser Zeichenkette codiert, um Speicherplatz zu sparen. So steht anstatt des ld ein Q in der Tabelle; hexadezimal 04 bedeutet r(G), 05 steht für r(H). G und H sind die oben eingeführten Variablen; bei r handelt es sich um eine Funktion, die in einer weiteren Tabelle entschlüsselt wird:

X	r(X)
0	B
1	C
2	D
3	E
4	H
5	L
6	(HL)
7	A

Die Funktion r bezeichnet also alle 8-Bit-Register der Z-80-CPU. Damit ergibt sich jetzt das echte Mnemonic: LD A,C

Es werden noch weitere Funktionen benötigt, von denen einigen ebenfalls Tabellen zugeordnet sind. Insgesamt gibt es sieben Tabellen dieser Art. Sie enthalten Bezeichnungen für Registerpaare, für Sprungbedingungen usw.

In ähnlicher Weise wie im obigen Beispiel werden alle Opcodes erkannt. Beginnt ein Befehl mit CB oder ED, so wird er in besonderer Weise bearbeitet, wofür wieder bestimmte Tabellenteile sorgen. Ein durch DD oder FD eingeleiteter Befehl wird zunächst so behandelt, als sei dieses einleitende Byte nicht vorhanden; erst ganz am Ende wird ix bzw. iy statt hl geschrieben und (ix+dd) bzw. (iy+dd) statt (hl), wobei sich dd direkt aus einem Byte des Befehls ergibt.

Das Fehlerrisiko ist klein

Die streng logische Vorgehensweise hat nur einen kleinen Schönheitsfehler. In bestimmten Situationen wird falsch disassembliert, wenn nämlich dem Disassembler unsinnige Bytefolgen vorgesetzt werden (diesen Fehler macht übrigens praktisch jeder Disassembler).

Ein typisches Beispiel: EB bedeutet *ex de,hl*. Der durch DDEB gegebene Befehl wird dann prompt als *ex de,ix* übersetzt, was auch logisch zu sein scheint. Die Z-80-CPU kann jedoch eine derartige Operation nicht ausführen; das Index-Byte DD wird in diesem Fall von ihr ignoriert und *ex de,hl* durchgeführt.

Natürlich könnte dieses Problem durch gezielte Abfragen auf derartige Sonderfälle gelöst werden, doch lohnt der Aufwand nicht, da solche unsinnigen Opcodes eigentlich gar nicht auftreten dürfen.

Michael Schramm

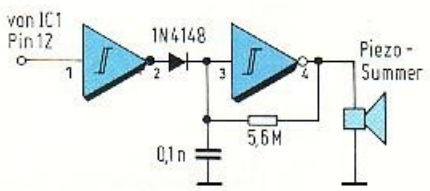
Verbesserungsvorschlag:

Tönender Schrittmacher

Wer sich den Schrittmacher für den ZX 81 nachgebaut hat (Auto-Repeat-Schaltung auf Seite 16), dem sind im IC CD 40106 noch zwei Schmitt-Trigger-Stufen übriggeblieben. Wird der Schrittmacher mit einer externen Tastatur verwendet, dann läßt er sich über diese beiden Stufen mit je einer Taste Ein- und Ausschalten (siehe auch Vorschlag auf Seite 64).

Die beiden Schmitt-Trigger können jedoch auch so geschaltet werden, daß sie einen Tastatur-Pieps ergeben, der jeden Tastendruck mit einem akkustischem Signal quittiert (Bild). Dabei wird der eine Schmitt-Trigger als Ton-Oszillator verwendet, der vom anderen Schmitt-Trigger freigegeben wird, wenn am Eingang (Pin 1) H-Pegel herrscht.

Wird als Lautsprecher ein Piezo-Schallgeber verwendet, paßt der Tastatur-Pieps auch noch komplett ins Gehäuse des ZX 81. Wichtig ist, daß beim „Nachrüsten" des Schrittmachers die zu Pin 1 und Pin 3 führenden Leiterbahnen aufgetrennt werden, da sonst die Schmitt-Trigger-Eingänge zwangsweise mit H-Pegel versorgt werden. M. Kiupel



Tastatur-Pieps: Diese Schaltung nutzt die übriggebliebenen Schmitt-Trigger des „Schrittmachers" von Seite 16

ZX-81-Hardwaretip:

Verriegelung für SHIFT

Insbesondere beim Erstellen von Grafiken mit dem ZX 81 vermißt man eine Shift-Lock-Taste. Sofern man eine externe Tastatur hat, kann man neben die Shift- oder A-Taste einen Schalter setzen und ihn parallel zur Shift-Taste anschließen.

Hat man nur die ZX-81-Tastatur, muß man auf diesen Komfort freilich auch nicht verzichten. Dann sind lediglich die Kontakte des Schalters mit den Tastaturanschlußpunkten KBD 0 und A 8 zu verbinden. Diese Anschlußpunkte sind Punkt 1 und Punkt 11 der Tastaturanschlußbuchsen auf der ZX-81-Platine.

Hans-Jürgen Ollech

Ausgabe von Morsezeichen:

Vom Klartext zum Morsecode

Das hier vorgestellte Programm ist das glatte Gegenstück zu dem auf Seite 51: Ein über die Tastatur des ZX 81 eingetippter Text wird auf Tastendruck komplett als Morsecode ausgegeben.

Das Programm stellt die Morsezeichen am MIC-Ausgang des ZX 81 bereit. Sie werden aber auch vom Lautsprecher des Fernsehgeräts wiedergegeben. Gebegeschwindigkeit und Längenverhältnisse zwischen Punkt, Strich und Pause können in einem sinnvollen Bereich frei gewählt werden. Da ein Maschinenprogramm die Erzeugung der Morsezeichen übernimmt, werden absolut exakte Zeichen gegeben.

Die Eingabe des Programms

Bild 1 zeigt das Basic-Listing. Es muß insbesonders bis zur Zeile 8 genau nach Vorlage eingetippt werden, damit sich die richtigen Adressen für den später einzulesenden Hex-Code ergeben. Hinter dem REM-Kommentar in Zeile 7 müssen genau 147 beliebige Buchstaben oder Ziffern eingetastet werden, hinter dem REM-Kommentar der nächsten Zeile 154 Zeichen. In den so reservierten Speicherplatz wird unter Zuhilfenahme eines Hex-Code-Eingabeprogramms (siehe Seite 72) der Hex-Code aus *Bild 2* und *Bild 3* untergebracht (Achtung: Startadressen anpassen!).

Nachdem das Programm einschließlich des Hex-Codes im Speicher steht, muß es noch an das ROM des vorliegenden ZX 81 angepaßt werden. Es gibt nämlich zwei verschiedene ROM-Versionen. Sinclair produziert schon seit längerem eine neue Version des ROMs, nachdem sich gezeigt hat, daß in der alten Version einige Schönheitsfehler stecken (Absturzgefahr nach einem PAUSE-Kommando im FAST-Modus, fehlerhaftes Rechnen in bestimmten Fällen).

Der Befehl PRINT PEEK 3880 ergibt 253, falls der ZX 81 mit einem ROM der alten Serie bestückt ist, und 59 bei der neuen Version. In der angegebenen Form funktioniert das Programm nur, sofern der ZX 81 über ein ROM der neuen Version verfügt. Alt-ROM-Besitzer müssen das Programm anpassen, indem sie die folgenden Befehle eintasten:

POKE 16670,32
POKE 16686,40
POKE 16723,67
POKE 16745,67
POKE 16750,40

Anschließend kann das Programm aufgezeichnet werden.

Vorbereitungen zum Morsen

Das Programm erwartet nach dem Start einige Eingaben.

○ Anzahl der Buchstaben pro Minute: Der Wert, der hier eingegeben wird, kann nur ungefähr eingehalten werden. Je nachdem, aus welchen Zeichen ein Text besteht und wie die Punkt-Strich-Pause-Verhältnisse gewählt werden, können sich erhebliche Abweichungen ergeben.

```
  1 REM COPYRIGHT BY
  2 REM FUNKSCHAU
  3 REM ZX-81-SONDERHEFT
  4 REM MICHAEL SCHRAMM
  5 REM
  6 PRINT "*MORSEN**"
  7 REM GENAU 147 ZEICHEN
  8 REM GENAU 154 ZEICHEN
 10 PRINT "WIEVIEL BUCHSTABEN J
E MINUTE?"
 20 INPUT G
 30 LET G=INT ABS G
 40 IF VAL "G>200 OR G<20" THEN
 GOTO VAL "20"
 50 PRINT G,,,,"VERHAELTNIS STR
ICHLAENGE ZU","PUNKTLAENGE (Z.B.
 3): ";
 60 INPUT V
 70 IF VAL "V<1.5 OR V>3" THEN
GOTO VAL "60"
 80 PRINT V,,,"VERHAELTNIS PUNK
TLAENGE ZU","PAUSENLAENGE (Z.B.
1): ";
 90 INPUT P
 100 IF VAL "P<.7 OR P>3" THEN G
OTO VAL "90"
 110 PRINT P,,,"TEXT FUER * EING
EBEN"
 120 INPUT S$
 130 LET K=VAL "1E4"/G
 140 LET A=VAL "16613"
 150 LET X=K
 160 GOSUB VAL "260"
 170 LET X=K*V
 180 GOSUB VAL "260"
 190 LET X=K/P
 200 GOSUB VAL "260"
 210 PRINT AT VAL "11",NOT PI;"T
EXT:"
 220 INPUT T$
 230 IF LEN T$ THEN LET M$=T$
 240 RAND USR 16669
 250 GOTO VAL "210"
 260 LET Y=INT VAL "X/256"
 270 POKE A,X-VAL "256"*Y
 280 POKE A+SGN PI,Y
 290 LET A=A+VAL "2"
 300 RETURN
```

① **Basic-Listing:** Es ist nur der „Wasserträger" für das Maschinenprogramm. Die VAL-Funktionen helfen Speicherplatz zu sparen

```
16619   47 4C 2D 6D 2A 35 31 22
16627   61 00 29 55 73 6A 3F 3E
16635   3C 3B 30 20 21 23 27 2F
16643   06 11 15 09 02 14 0B 10
16651   04 1E 0D 12 07 05 0F 16
16659   1B 0A 08 03 0C 18 0E 19
16667   1D 13 CD 23 0F 11 00 08
16675   10 FE 1B 7A B3 20 F9 CD
16683   7E 41 CD 2B 0F C9 2A 10
16691   40 7E B9 20 06 23 5E 23
16699   56 23 C9 FE 80 CB 11 06
16707   00 E6 20 CC 3B 41 19 1B
16715   EB D3 FF 06 4B 10 FE CD
16723   46 0F 30 16 06 43 10 FE
16731   2B 7C B5 20 EC C9 2A E9
16739   40 06 BD 10 FE CD 46 0F
16747   3B 05 CD 2B 0F CF 0C 2B
16755   7C B5 20 ED C9
```

② **Hex-Code für Zeile 7:** Die FOR-NEXT-Schleife eines Eingabe-Programms muß für diese Codes von Adresse 16619 bis einschließlich Adresse 16759 reichen

○ Verhältnis Strichlänge zu Punktlänge: Bei geringer Gebegeschwindigkeit ist der Wert 3 üblich; um sehr hohe Geschwindigkeiten zu erreichen, ist jedoch ein kleinerer Wert – aber größer als 2 – für den Hörer günstiger.

```
16766   0E 52 CD 31 41 FE B0 CB
16774   7A B3 CB ED 53 32 40 22
16782   25 40 7E E6 7F CB 77 20
16790   74 A7 20 0A 0E 02 CD 61
16798   41 0D 20 FA 18 67 FE 0C
16806   38 63 28 D4 FE 17 20 11
16814   D5 E5 0E 58 CD 80 41 E1
16822   22 25 40 E1 22 32 40 18
16830   4C FE 0D 20 11 2B 7E 1E
16838   06 FE 26 2B 2F 1E 02 FE
16846   34 28 29 1C 18 26 FE 28
16854   20 14 23 47 7E FE 2D 78
16862   20 0C 22 25 40 1B ED 53
16870   32 40 1E 1F 18 08 21 DD
16878   40 06 00 4F 09 5E CD 61
16886   41 CD 61 41 2A E5 40 CB
16894   38 28 0A 30 03 2A E7 40
16902   CD 4C 41 18 EC ED 5B 32
16910   40 1B 2A 25 40 23 C3 86
16918   41
```

③ **Hex-Code für Zeile 8:** Die Adressen dieser Codes reichen von 16766 bis 16918

○ Verhältnis Punktlänge zu Pausenlänge: Üblich ist 1; für höchste Geschwindigkeiten kann jedoch ein etwas größerer Wert gewählt werden.

○ Text für ∗: Hier kann ein Standardtext eingegeben werden, der später bei der Morsetext-Eingabe dann einfach durch Eintippen eines Sternchens aufgerufen wird. Der Sternchen-Text selbst darf kein Sternchen enthalten.

○ Text: Nun wird der zu morsende Text eingetastet. Ein Sternchen wird dann später beim Morsen durch den zuvor zugeordneten Standardtext ersetzt. Drückt man nur NEWLINE, so wird der zuletzt eingegebene Text nochmals gemorst. Die Umlaute Ä, Ö und Ü können, falls gewünscht, durch AS, O$ bzw. U$ dargestellt werden. CH wird automatisch als ein Zeichen erkannt (viermal lang); falls man es wünscht, daß C und H beim Morsen nicht zu einem Zeichen zusammengezogen werden, muß man vor dem Programmstart POKE 16854,24 eingeben.

Ein Pfund-Zeichen am Ende des Eingabetextes bewirkt dessen ständige Wiederholung; das Morsen kann jedoch jederzeit durch Drücken der BREAK-Taste abgebrochen werden. Noch einige weitere Sonderzeichen haben Bedeutung für das Morsen: + warten; < Spruchanfang; > Spruchende.

④ **Assembler-Listing:** Die Adressen (linke Spalte) und die Sprungziele relativer Sprünge sind hexadezimal dargestellt (Dollar-Markierung); die übrigen Werte dezimal

```
ZUNAECHST TABELLE IN ZEILE 7

16619   47 4C 2D 6D 2A 35 31 22
16627   61 00 29 55 73 6A 3F 3E
16635   3C 3B 30 20 21 23 27 2F
16643   06 11 15 09 02 14 0B 10
16651   04 1E 0D 12 07 05 0F 16
16659   1B 0A 08 03 0C 1B 0E 19
16667   1D 13

ECHTER MASCHINENCODE IN ZEILE 7

411D CALL 03875          CD230F
4120 LD DE,02048         110008
4123 DJNZ $4123          10FE
4125 DEC DE              1B
4126 LD A,D              7A
4127 OR E                B3
4128 JR NZ,$4123         20F9
412A CALL 16766          CD7E41
412D CALL 03883          CD2B0F
4130 RET                 C9

4131 LD HL,(16400)       2A1040
4134 LD A,(HL)           7E
4135 CP C                B9
4136 JR NZ,$413E         2006
413B INC HL              23
4139 LD E,(HL)           5E
413A INC HL              23
413B LD D,(HL)           56
413C INC HL              23
413D RET                 C9
413E CP 128              FE80
4140 RET Z               C8
4141 LD DE,00006         110600
4144 AND 032             E620
4146 CALL Z,16696        CC3841
4149 ADD HL,DE           19
414A JR $4134            18E8

414C OUT 255,A           D3FF
414E LD B,072            0648
4150 DJNZ $4150          10FE
4152 CALL 03910          CD460F
4155 JR NC,$416D         3016
4157 LD B,067            0643
4159 DJNZ $4159          10FE
415B DEC HL              2B
415C LD A,H              7C
415D OR L                B5
415E JR NZ,$414C         20EC
4160 RET                 C9
4161 LD HL,(16617)       2AE940
4164 LD B,141            068D
4166 DJNZ $4166          10FE
4168 CALL 03910          CD460F
416B JR C,$4172          3B05
416D CALL 03883          CD2B0F
4170 RST 08              CF
4171 DFB 12              0C
4172 DEC HL              2B
4173 LD A,H              7C
4174 OR L                B5
4175 JR NZ,$4164         20ED
4177 RET                 C9

MASCHINENCODE IN ZEILE 8

417E LD C,082            0E52
4180 CALL 16689          CD3141
4183 CP 128              FE80
4185 RET Z               C8

4186 LD A,D              7A
4187 OR E                B3
4188 RET Z               C8
```

FAST-Modus einschalten.
Einige Sekunden warten, bevor die Morsezeichen gesendet werden.

Morseprogramm aufrufen.
SLOW-Modus einschalten.
Rücksprung ins Basic.

HL:= (VARS).
Variablencodierung in Akku.
Gesuchte Variable steht in C.
Falls nicht gleich, weitersuchen.
HL auf Länge der String-Variablen.
Länge des Strings in DE laden.

HL auf Stringausdruck setzen.
Rücksprung.
Var.-Speicher-Ende erreicht?
Falls ja, Rücksprung.
Länge einer normalen Variablen.
Falls String-Variable vorliegt, Länge feststellen.
HL auf nächste Variable.
Diese Variable testen.

Einen Ton mit HL-Perioden am Kassettenrecorder-Ausgang MIC erzeugen; dabei ständig abfragen, ob die SPACE-Taste gedrückt ist.

Länge einer kurzen Pause in HL.
Kurze Pause erzeugen; auch hierbei prüfen, ob die SPACE-Taste gedrückt ist.
SLOW-Modus einschalten.
Error-Routine aufrufen mit Meldung D.

Codierung für die Var. M$.
M$ im Var.-Speicher suchen.
Abbruch, falls nicht vorhanden.

Falls keine Zeichen mehr zu erzeugen sind, Rücksprung.

```
4189 LD (16434),DE      ED533240   Die Anzahl merken.
418D LD (16421),HL      222540     Aktuelle Adresse in M$ merken.
4190 LD A,(HL)          7E         Zu morsendes Zeichen in A.
4191 AND 127            E67F       Invers zu normal machen.
4193 BIT 6,A            CB77       ZX-Schlüsselwort?
4195 JR NZ,$420B        2074       Falls ja, überlesen.
4197 AND A              A7         SPACE?
419B JR NZ,$41A4        200A       Falls nein, weiter.
419A LD C,002           0E02       Falls ja, nur zwei kurze
419C CALL 16737         CD6141     Pausen erzeugen.
419F DEC C              0D
41A0 JR NZ,$419C        20FA
41A2 JR $420B           1867

41A4 CP 012             FE0C       Grafik-Zeichen?
41A6 JR C,$420B         3863       Falls ja, überlesen.
41A8 JR Z,$417E         28D4       Morsen wiederholen, falls £.
41AA CP 023             FE17       Sternchen? Falls nein, weiter.
41AC JR NZ,$41BF        2011       DE auf den Stack legen.
41AE PUSH DE            D5         HL auf den Stack legen.
41AF PUSH HL            E5         Codierung für die Var. S$.
41B0 LD C,08B           0E58       Inhalt von S$ morsen.
41B2 CALL 16768         CDB041     HL wiederherstellen.
41B5 POP HL             E1
41B6 LD (16421),HL      222540     Den alten Inhalt von DE
41B9 POP HL             E1         wieder in 4032h laden.
41BA LD (16434),HL      223240     Nächstes Zeichen nach *.
41BD JR $420B           184C       Dollar? Falls nein, weiter.
41BF CP 013             FE0D       Zuletzt gemorstes Zeichen
41C1 JR NZ,$41D4        2011       untersuchen. Umlaut-Code für A.
41C3 DEC HL             2B         Letztes Zeichen = A?
41C4 LD A,(HL)          7E         Falls ja, Umlaut-Endung morsen.
41C5 LD E,006           1E06       Umlaut-Code für O. Letztes Zeichen = O?
41C7 CP 038             FE26       Dann Umlaut-Endung morsen.
41C9 JR Z,$41FA         282F       Umlaut-Code für U.
41CB LD E,002           1E02       Umlaut-Endung morsen.
41CD CP 052             FE34
41CF JR Z,$41FA         2829
41D1 INC E              1C
41D2 JR $41FA           1826

41D4 CP 040             FE28       Buchstabe C?
41D6 JR NZ,$41EC        2014       Falls nein, weiter.
41D8 INC HL             23         Falls ja, prüfen, ob das
41D9 LD B,A             47         nächste Zeichen ein H ist.
41DA LD A,(HL)          7E
41DB CP 045             FE2D
41DD LD A,B             78
41DE JR NZ,$41EC        200C       Bei positivem Ergebnis
41E0 LD (16421),HL      222540     CH als ein Zeichen
41E3 DEC DE             1B         verarbeiten.
41E4 LD (16434),DE      ED533240   Morsecodierung für CH.
41E8 LD E,031           1E1F
41EA JR $41F4           1808

41EC LD HL,16605        21DD40     Morsecodierung
41EF LD B,000           0600       des Zeichens aus
41F1 LD C,A             4F         der Tabelle
41F2 ADD HL,BC          09         in Register E
41F3 LD E,(HL)          5E         holen.
41F4 CALL 16737         CD6141     Kurze Pause vor dem Zeichen.
41F7 CALL 16737         CD6141     Vor jedem Bit eine Pause.
41FA LD HL,(16613)      2AE540     Punktlänge in HL.
41FD SRL E              CB3B       Ein Bit ausrotieren.
41FF JR Z,$420B         280A       Wenn Rest = 0, dann fertig.
4201 JR NC,$4206        3003       0-Bit bedeutet Punkt.
4203 LD HL,(16615)      2AE740     Strichlänge in HL.
4206 CALL 16716         CD4C41     Ton erzeugen.
4209 JR $41F7           18EC       Nächstes Bit.

                                   Anzahl noch zu morsender
420B LD DE,(16434)      ED5B3240   Zeichen minus Eins.
420F DEC DE             1B         HL auf nächste Adresse
4210 LD HL,(16421)      2A2540     im String.
4213 INC HL             23         Das Zeichen morsen.
4214 JP 16774           C38641
```

Gemorst wird im FAST-Modus

Die Morsezeichen stehen am MIC-Ausgang des ZX 81 zur Verfügung und können auf Band aufgezeichnet oder durch einen Verstärker wiedergegeben werden. Außerdem sind die Töne im Fernsehlautsprecher zu hören, allerdings je nach verwendetem Fabrikat mehr oder weniger verbrummt; die Kanalabstimmung muß unter Umständen leicht verstellt werden.

Leider kann der ZX 81 im SLOW-Modus keine reinen Töne erzeugen, daher wird im FAST-Modus gemorst; auf dem Bildschirm erscheint dabei ein Streifenmuster (ähnlich dem SAVE-Bild).

Die Frequenz des Tonsignals läßt sich verändern, indem man mit POKE den Adressen 16719 und 16728 neue Werte zuordnet (zwischen 70 und 255; 255 für den tiefstmöglichen Ton), und zwar beiden Adressen denselben Wert. Um zu erreichen, daß dann weiterhin mit der gewählten Geschwindigkeit und den gewählten Punkt-Strich-Pause-Verhältnissen gemorst wird, sind im Basic-Programm zwei Zeilen zu erweitern:

150 LET X=K/a
170 LET X=K*V/a

Hierbei muß a durch eine Zahl zwischen etwa 1 und 3.6 (in beiden Zeilen gleich) ersetzt werden. Den genauen Wert für a ermittelt man am besten experimentell.

So arbeitet das Morseprogramm

Der Basic-Teil dient nur dazu, die erforderlichen Eingaben zu verlangen und entgegenzunehmen. Das eigentliche Morsen vollzieht sich im Maschinencode-Teil des Programms. Hierbei spielt der Hex-Code in den Zeilen 7 und 8 eine wichtige Rolle.

○ Zeile 7: je 2 Byte für Punkt-, Strich- und Pausenlänge (Periodenzahl). 50 Byte Morsezeichentabelle (Anfangsadresse 16619). 91 Byte Maschinensprache-Unterprogramme.

○ Zeile 8: das Maschinencode-Hauptprogramm mit 154 Byte Umfang.

Besonders interessant ist der Aufbau der Tabelle, die die Morse-Codierungen der 50 Zeichen vom Doppelpunkt bis Z

(Reihenfolge im ZX-81-Code) enthält. Ein Beispiel zeigt, wie die Codierung vorgenommen wurde.

Das Byte mit Adresse 16654 hat den Inhalt 18. Das ist binär dargestellt: 0 0 0 1 0 0 1 0. Die Bits werden von rechts (das heißt ab Bit 0) gelesen; 0 bedeutet Punkt, 1 steht für Strich. Die letzte 1 hat nur die Funktion einer Ende-Markierung und wird nicht umgesetzt. Diese Markierung ist erforderlich, weil die Morsezeichen unterschiedlich lang sind. Somit ergibt sich Punkt-Strich-Punkt-Punkt, also die Codierung des Buchstabens L. Entsprechend sind alle Zeichen in der Tabelle codiert.

Die Morseroutine liest für jedes zu erzeugende Morsezeichen seine Codierung aus der Tabelle und erzeugt daraus die richtigen Ton-Pause-Folgen. Dieses geschieht durch schnelle Sequenzen von abwechselnd *in*- und *out*-Befehlen im Maschinencode. Für Kenner der Z-80-Maschinensprache zeigt *Bild* 4 das kommentierte Assemblerlisting des Programms.

Michael Schramm

ZX-81-Softwaretip:

Zwei Zeilen mehr

Normalerweise kann man beim ZX 81 nur 22 Zeilen des Bildschirms mit PRINT ansprechen, die zwei untersten Zeilen sind für Eingaben (INPUT) und Fehlermeldungen reserviert. Durch Verändern der Systemvariablen DF-SZ, deren normaler Wert 2 ist, läßt sich dies ändern. Setzt man DF-SZ nämlich auf 0, so darf man alle 24 Bildschirmzeilen direkt ansprechen, was bei Spielen oder längeren Texten sehr nützlich sein kann.

Beim ZX 81 ist DF-SZ unter der Adresse 16418 gespeichert, man muß also an geeigneter Stelle in seinem Programm den Befehl POKE 16418,0 einbauen. Der ZX 81 setzt vor Fehlermeldungen oder am Programmende DF-SZ automatisch wieder auf Zwei zurück.

Nur vor der Anwendung von SCROLL oder INPUT muß man dies selber tun, um einen Absturz zu vermeiden. Was im unteren Bildschirmbereich geschrieben wurde, bleibt trotzdem erhalten, solange es nicht überschrieben wird.

Wolf-Dieter Roth

ZX-81-Software:

Zahlenkolonnen in den Griff bekommen

Die Eingabe mehrstelliger Zahlenreihen (z. B. Berechnung zur Bahnverfolgung von Amateurfunk-Satelliten, Reihenauswertung von elektrischen Meßdaten, statistische Übersichten) über die Tastatur des ZX 81 ist sehr zeitraubend.

Mit nachfolgendem Programm (*Bild 1*) können solche Zahlenreihen in verkürzter Form eingegeben werden, was die Eingabezeit nach einiger Übung erheblich vermindert und die Fehlermöglichkeit stark herabsetzt (A, B, C, D, S und X sind die verwendeten Variablen).

Das Programm beginnt mit der Eingabe einer ganzstelligen Zahl X (Anfangswert) in Zeile 40. Zeile 60 speichert das Ergebnis der Vorzeichenfunktion SGN in D ab (z. B. −1 wenn X negativ ist). Nun erfolgt in Zeile 80 die Eingabe der Zahl A (neuer Wert), jedoch in verkürzter Form (siehe Eingabebeispiel). Daran anschließend werden mit Zeile 90 und 100 die in X und A eventuell negativ gespeicherten Zahlen in positive umgewandelt, um in Zeile 130 fehlerfrei logarithmieren zu können.

In Zeile 110 wird zur Eingabe eines neuen Anfangswertes X (Zeile 40) verzweigt, wenn in Zeile 80 Null eingegeben wurde. Dabei werden die alten Zahlenwerte auf dem Bildschirm gelöscht (Zeile 220 und 230) und der Inhalt von D (Vorzeichen) auf Null zurückgesetzt (Zeile 30). In den weiteren Programmzeilen erfolgt die eigentliche Modifikation der Zahl X durch die verkürzt eingegebene Zahl A.

Zunächst wird in Zeile 120 der Inhalt von A nach B übertragen, um in B Zwischenergebnisse abspeichern zu können. Zeile 130 ermittelt schließlich die Anzahl der Stellen, welche durch die Zahl A ersetzt werden (die Kennzahl des Logarithmus der Zahl A ist gleich der um 1 verminderten Anzahl der Stellen).

Die Zeilen 140 bis 180 übertragen alle Stellen der Zahl A rechtsbündig in die Zahl X. Der Inhalt von X ist somit der Neue Wert. Zuletzt wird die Zahl X in Zeile 190 mit D multipliziert und erhält somit wieder das ursprüngliche Vorzeichen.

Nachdem die Zahl X (neuer Wert) zur Summierung in die Variable S übertragen wurde, geht das Programm zur Anzeige des neuen Wertes zur Zeile 200 und dann zur Eingabe der nächsten Zahl A zur Zeile 80 zurück. Die Summe aller eingegebenen Zahlen kann am Ende des Programms (Unterbrechen mit STOP) mit dem Befehl PRINT S zur Anzeige gebracht werden. Für die weitere Bearbeitung in einem Hauptprogramm stehen folgende Variablen zur Verfügung: S = Summe aller Zahlen; X = jeweils aktuelle Zahl.

Anton Weiß

```
 10 REM GEKUERZTE ZAHLENEINGABE
 20 PRINT AT 5,3;"ANFANGSWERT";
 30 LET D=0
 40 INPUT X
 45 LET S=X
 50 PRINT AT 5,17;X
 60 LET D=SGN X
 70 PRINT AT 7,4;"NEUER WERT:";
 80 INPUT A
 90 LET X=ABS X
100 LET A=ABS A
110 IF A=0 THEN GOTO 220
120 LET B=A
130 LET B=INT (LN B/LN 10)+1
140 LET C=1
150 FOR N=1 TO B
160 LET C=C*10
170 NEXT N
180 LET X=INT (X/C)*C+A
190 LET X=X*D
195 LET S=S+X
200 PRINT AT 7,17;X;"     ";
210 GOTO 80
220 PRINT AT 5,17;"          "
230 PRINT AT 7,17;"          "
240 GOTO 30
```

Eingabebeispiel:

ANFANGSWERT: 455341.88
NEUER WERT:

(Eingabe)	(Anzeige)
82.09	455382.09
124.23	455124.23
1.03	455121.03
399878.11	399878.11

① **Programmlisting für ZX 81:** Wichtig ist, daß die Zeichenketten mit Leerzeichen genau nach Vorlage eingegeben werden

ZX-81-Grundlagen:

Schiebung im Speicher

Die einzelnen Speicherbereiche des ZX 81 sind so bemessen, daß die Informationen genau hineinpassen (siehe Sinclair-Handbuch, Kapitel 27). Fügt man z. B. in ein Programm eine Programmzeile LET A=1000 ein, so wird der Programmspeicher erweitert, indem die darüberliegenden Bereiche zum Speicherende (RAMTOP) hin verschoben werden. Dadurch verkleinert sich der noch vorhandene freie Speicherplatz.

Im ersten Teil des Speichers stehen die Systemvariablen. Mit Hilfe dieser Informationen erkennt der Rechner u. a., wo die Grenzen der einzelnen Speicherbereiche sind. Durch Abfrage der Systemvariablen D-FILE (Adresse 16396 und 16397) erhält man z. B. die Adresse des Bildspeicheranfangs.

Zur vereinfachten Darstellung der Speicherverteilung wird der RAM-Speicher in nur noch drei Abschnitte unterteilt (Programm-, Bild- und Variablenspeicher), und es wird davon ausgegangen, daß ein 16-KByte-RAM-Modul angeschlossen ist. Jede eingegebene Programmzeile vergrößert auch jetzt den Programmspeicher und verschiebt somit den Beginn von Bild- und Variablenspeicher. Für den Bildspeicher hat das Betriebssystem des ZX 81 genau 793 Byte reserviert, die einen kompletten Bildschirminhalt mit allen dazu benötigten Informationen speichern.

Solange noch keine Programmzeile eingegeben wurde, beginnt der Bildspeicher hinter den Systemvariablen bei Adresse 16509. Da 793 Byte für ihn reserviert sind, endet er bei Adresse 17302. Variablen wurden auch nicht definiert, so daß dieser Bereich bei Adresse 17302 endet. Durch Eingabe des Programms (*Bild*) kann man nun die Verschiebung dieser Speicherbereiche selbst beobachten.

Da das Programm 284 Byte lang ist, beginnt der Bildspeicher jetzt bei Adresse 16793 und er endet bei Adresse 17586. Und weil beim ersten GOSUB-Aufruf die Variable A noch nicht definiert ist, ist der Variablenspeicher leer.

Nach dem zweiten GOSUB-Aufruf hat der Variablenspeicher dagegen eine Länge von 6 Byte. Da der ZX 81 Zahlen intern stets mit 6 Byte speichert, bleibt die Länge des Variablenspeichers auch gleich, wenn jetzt der Variablen A z. B. der Wert 1 zugewiesen wird; das verkürzt nur den Programmspeicher! Weisen Sie der Variablen A auch einmal negative Werte zu oder definieren Sie mit DIM Felder, und beobachten Sie die Auswirkungen.

Wolfgang Lüdtke

```
  1 GOSUB 5
  2 LET A=1000
  3 GOSUB 5
  4 STOP
  5 PRINT ,,16509;"  PS ANFANG"
 10 PRINT PEEK 16396+256*PEEK 1
6397;"   PS ENDE; BS ANFANG"
 20 PRINT PEEK 16400+256*PEEK 1
6401;"   BS ENDE; VS ANFANG"
 30 PRINT (PEEK 16404+256*PEEK
16405)-1;"   VS ENDE"
 50 RETURN
```

Übungsprogramm: Die Variablenzuweisung in Zeile 2 hat merklichen Einfluß auf die Speicherverteilung

ZX-81-Hardwaretip:

Einfühlsamer Sensor-SHIFT

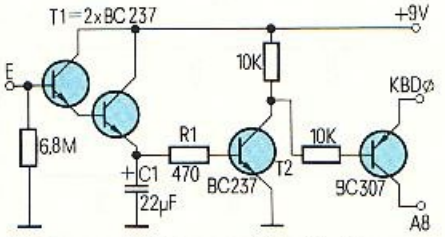
Insbesondere bei der SHIFT-„Taste" der ZX-81-Folientastatur macht sich das Fehlen eines Druckpunkts störend bemerkbar. Wie schnell hat man diese Taste verfehlt und damit ein falsches Zeichen eingegeben. Diesen Mangel beseitigt eine Sensor-SHIFT-Taste. Sie hat außerdem in der beschriebenen Ausführung den Vorteil, daß SHIFT und eine weitere Taste mit einer Hand zu betätigen sind. Das erleichtert z. B. die Ausführung der Löschfunktion (SHIFT und RUBOUT).

Die Schaltung der einfachen Sensor-Elektronik zeigt *Bild 1*. Sobald man mit einem Finger den Basiskontakt von T1 berührt, steuert die positive Halbwelle des in fast jedem Raum vorhandenen 50-Hz-Feldes diesen Darlington-Transistor durch. Damit wird C1 aufgeladen und über R1 auch T2 durchgesteuert, der wiederum den BC 307 öffnet. Dann ist die SHIFT-Taste (Anschlüsse A8 und KBD 0) durch die Kollektor-Emitter-Strecke des BC 307 überbrückt, und die Funktion wird ausgeführt. Kommt es dabei zu Unterbrechungen (flattern), ist der Wert von C1 höher zu wählen, und bleibt SHIFT nach Loslassen des Sensors kurz „kleben", dann ist C1 kleiner zu dimensionieren. Die gesamte Schaltung paßt noch ins Gehäuse des ZX 81.

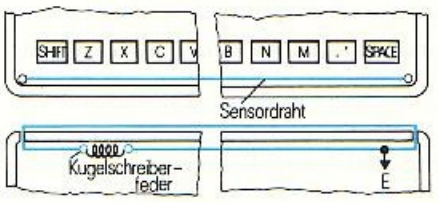
Der Anschluß KBD 0 ist der erste am fünfpoligen Klemmsockel für die Folientastatur. Das Signal A8 ist an der Diode D6 des daneben angeordneten achtpoligen Klemmanschlusses verfügbar. Kennzeichnungen auf der Platine helfen beim Aufspüren der beiden Anschlüsse.

Für den metallischen Sensor bietet sich die in *Bild 2* gezeigte Lösung an. Hier ist ein als Sensor wirkender Draht über die gesamte Breite des ZX-81-Tableaus gespannt. Dieser Draht läßt sich auch blind sofort ertasten.

Hartmut Heinz



① **Sensor-Elektronik:** Ein Berühren von Punkt E bringt den BC 307 zum Durchschalten



② **Montage des Sensors:** Blanker Draht spannt sich über die gesamte Breite des ZX-81-Tastenfeldes

ZX 81 à la carte (3)

Software voller Klang

Wer die Grundplatine zu dieser Serie nachgebaut hat, kann jetzt allein mit Software dem ZX 81 Töne entlocken. Selbst eine Miniorgel läßt sich so verwirklichen.

Die in den beiden FUNKSCHAU-Heften 12 und 13 (1983) vorgestellte Grundplatine zu dieser Serie trägt auch einen softwaregesteuerten Schalter zum Betrieb eines Lautsprechers. Mit Hilfe von Programmen ist der ZX 81 nun in der Lage, Töne zu erzeugen. Dadurch lassen sich z. B. akustische Marken in einen Programmablauf einbauen. Sogar eine elektronische Miniorgel läßt sich „programmieren".

Basic setzt der Tonhöhe Grenzen

Das Prinzip der Tonerzeugung ist einfach: Der Software-Schalter wird durch das Steuerprogramm lediglich in schneller Folge umgeschaltet. Da es sich bei diesem Schalter um ein Set-Reset-(SR-)Flipflop handelt, läßt sich damit das Tastverhältnis beliebig einstellen (*Bild 1*). Das folgende kleine Programm erzeugt bereits einen Ton:

```
10 POKE 23556,0
20 POKE 23557,0
30 GOTO 10
```

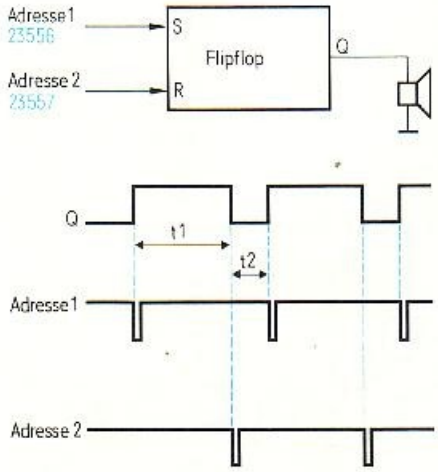
Befindet sich der ZX 81 im SLOW-Modus, so klingt der Ton ein wenig jaulend. Der Grund dafür ist, daß der Computer 50mal pro Sekunde die Programmabarbeitung unterbricht, um ein Fernsehbild darzustellen. Wird auf den FAST-Modus umgeschaltet, fehlt das Jaulen und der Ton klingt höher: Eine definierte Tonerzeugung ist also nur im FAST-Modus möglich.

Es ist unschwer zu erkennen, daß die Tonfrequenz von den Zeiten t1 und t2 (Bild 1) bestimmt wird. Für die Grundfrequenz der stark oberwellenhaltigen Schwingung gilt: $f = 1/(t1 + t2)$. Die Frequenz wird also durch die Zeit bestimmt, die zum Abarbeiten der Programmbefehle benötigt wird. Man kann das leicht testen, indem man zwischen die POKE-Befehle weitere (beliebige) Befehle einschiebt, z. B.:

```
15 LET A = 0
25 LET A = 0
```

Die Frequenz wird dadurch niedriger. Im SLOW-Modus wird zusätzlich Zeit für die Bildausgabe benötigt; deshalb ist die Tonhöhe dann stets geringer als im FAST-Modus.

Will man die Tonfrequenz erhöhen, so erkennt man sehr schnell, daß das in Basic nicht geht: Nach dem Löschen der Zeilen 15 und 25 hat man bereits das schnellstmögliche Basicprogramm zur Tonerzeugung. Ein weiteres Steigern der Tonhöhe ist nur mit einem Maschinenprogramm anstelle des Basic-Programms möglich. Maschinenprogramme werden erheblich schneller abgearbeitet als Basic-Programme.



① **Software-Schalter:** Unter den beiden Adressen erteilte Set-Reset-Impulse bestimmen das Tastverhältnis des Tonsignals am Ausgang Q

```
100 REM 0000000000000000000000
        0000000000000000000000
110 REM S-E: PROGRAMM POKEN
120 PRINT "START : ";
130 INPUT S
140 PRINT S;" ENDE : ";
150 INPUT E
160 PRINT E
170 FOR N=S TO E
180 PRINT N;
190 INPUT K
200 POKE N,K
210 PRINT " - "; PEEK N
220 NEXT N
230 STOP
300 REM S-E: SPEICHERINHALT
310 PRINT "START : ";
320 INPUT S
330 PRINT S; " ENDE : ";
340 INPUT E
350 PRINT E
360 FOR N=S TO E
370 PRINT N;" - ";PEEK N
380 NEXT N
390 STOP
```

② **Hilfsprogamm:** Dieses Programm dient der (dezimalen) Eingabe von Maschinenprogrammen und zu deren Kontrolle

Tonerzeugung per Maschinenprogramm

Mit der USR-Funktion läßt sich beim ZX 81 ein zuvor geschriebenes Maschinenprogramm aufrufen. Zum Einschreiben eines solchen Programms in das RAM gibt es verschiedene Methoden (siehe auch Sinclair-Handbuch Kapitel 26).

Für unser Vorhaben ist die Unterbringung in einem (an erster Stelle im Programm stehenden) REM-Kommentar zweckmäßig. Hierzu werden nach dem REM-Befehl mindestens so viele (beliebige) Zeichen eingetippt, wie das spätere Maschinenprogramm Befehle hat. Anschließend wird an die Stelle der Zeichen das gewünschte Maschinenprogramm mit POKE-Befehlen eingeschrieben.

Da Maschinenprogramme jetzt öfters benötigt werden, ist es sinnvoll, das Hilfsprogramm aus *Bild 2* einzutippen und abzuspeichern. Es enthält neben dem „Platzhalter" für das Maschinenprogramm (Zeile 100) eine Hilfsroutine zum Poken (Zeile 120...230) und zum Kontrollieren (Zeile 300...390; Aufruf: RUN 300) des Maschinenprogramms.

Mit Hilfe des POKE-Programms kann sofort ein kleines Maschinenprogramm zur Tonerzeugung geladen werden (*Bild 3*). Startadresse ist 16514 (Adresse im Speicher, in der der Code für das erste Zeichen im REM-Kommentar abgelegt ist) und die Endadresse ist 16533. Aufgeführt ist die jeweilige Speicheradresse, sowie die Befehlsangabe in Assembler- und Maschinensprache (dezimal).

Ist der letzte Maschinenbefehl eingegeben worden, meldet sich der Rechner mit 9/230 (bei Bildschirmüberlauf weiter mit CONT). Nach Drücken der CONT-Taste läßt sich das Maschinenprogramm überprüfen.

Der Inhalt des REM-Kommentars hat sich jetzt verändert. Die nunmehr dargestellten Symbole entsprechen den Maschinenbefehlen. Nun muß das Programm noch um die Zeile

400 LET A = USR 16514

ergänzt werden (Aufruf des Maschinenprogramms aus dem Basic-Programm).

Bevor das Programm mit RUN 400 (FAST-Modus) gestartet wird, sollte es unbedingt abgespeichert werden. Das erspart viel Tipp-Arbeit, falls das Maschinenprogramm infolge eines Eingabefehlers „abstürzen" sollte. Wenn man das Hilfsprogramm nicht mehr benötigt, darf man es bis auf die Zeile 100, die ja das Maschinenprogramm enthält, aus dem RAM tilgen.

Adresse	Mnemonik	Eingabe (dezimal)
16514	ld a,255	62
16515		255
16516	ld b,255	6
16517		255
16518	djnz,−2	16
16519		254
16520	ld (23556),a	50
16521		4
16522		92
16523	ld b,255	6
16524		255
16525	djnz,−2	16
16526		254
16527	ld (23557),a	50
16528		5
16529		92
16530	dec a	61
16531	jrnz,−17	32
16532		239
16533	ret	201

③ **Tonerzeugung:** Das Maschinenprogramm erzeugt Töne deutlich höherer Frequenz, als es ein Basic-Programm vermag

Ladebefehle ersetzen POKE-Befehle

Wie *Bild 4* zeigt, ist die Tonerzeugung in Maschinensprache ähnlich aufgebaut wie in Basic: Den POKE-Befehlen von dort entsprechen die Ladebefehle unter den Adressen 16520 bzw. 16527 (*ld (NN),a*) zum Umschalten des Flipflops.

Da das Maschinenprogramm sogar zu schnell ist, müssen zwischen die Schaltbefehle noch Programmschleifen zur Verzögerung eingebaut werden. Hierbei wird der Inhalt des b-Registers bis auf 0 dekrementiert (heruntergezählt). Der entsprechende Anfangswert des b-Registers (Speicherstelle 16517 bzw. 16524) bestimmt die hierfür nötige Zeit und damit die Frequenz.

Nach jeder Schwingungsperiode wird der Akku-Inhalt dekrementiert, sein Anfangswert (Speicherstelle 16515) bestimmt also die Tondauer (genauer: die Anzahl der Schwingungszüge, die Tondauer ist also frequenzabhängig).

Ein Berechnen der Frequenz und Tondauer ist im Prinzip möglich, wenn man die Zeiten aufaddiert, die zur Abarbeitung der einzelnen Maschinenbefehle benötigt werden. Da die Z-80A-CPU mit einer Taktfrequenz von 3,23 MHz arbeitet, dauert z. B. das Abarbeiten des dec-Befehls 1,24 µs. Weil aber die bedingten Sprungbefehle (*jrnz, djnz*) unterschiedlich lange dauern, je nachdem ob die Bedingung erfüllt ist oder nicht, ist dieses Verfahren nicht ganz einfach; Experimentieren führt meist schneller zum Ziel.

Basic-Programme rufen das Maschinenprogramm

Akustische Marken im Ablauf eines Programmes haben einen hohen Aufmerksamkeitswert. Dabei ist es besonders günstig, wenn das hierzu benutzte Maschinenprogramm überhaupt nicht in Erscheinung tritt, sondern „versteckt" wird.

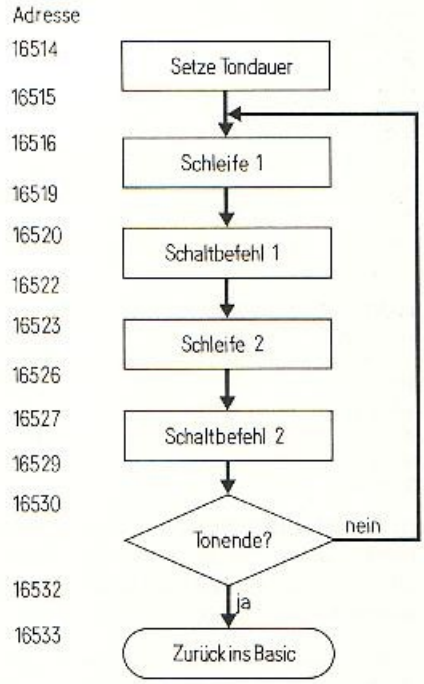
Dies läßt sich erreichen, indem man den Wert der Systemvariablen RAMTOP herabsetzt, der Computer „vergißt" dadurch gewissermaßen den oberhalb von RAMTOP verbliebenen Teil seines Speichers. In diesem vergessenen Speicherteil wird anschließend das Maschinenprogramm untergebracht. Wenn die 6-KByte-Erweiterung voll bestückt ist, hat RAMTOP den Wert 23552. Durch

POKE 16388, 204 und
POKE 16389, 91

wird RAMTOP auf 23500 herabgesetzt. Anschließend muß man NEW eingeben und danach das Tonerzeugungsprogramm vom Band laden. Ihm wird dann noch folgendes Programm angefügt:

```
400 FOR N=16514 T0 16533
410 POKE (N+6986), PEEK N
420 NEXT N
```

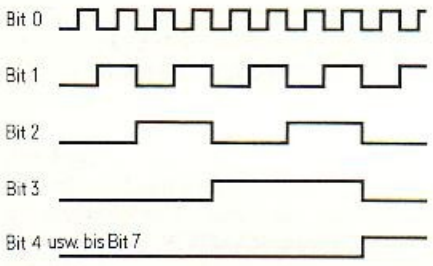
Startet man diesen Programmteil mit RUN 400, so kopiert er das Maschinenprogramm in den Speicherbereich oberhalb von RAMTOP. Das Maschinenprogramm ist dort ohne Probleme lauffähig, da es nur relative Sprünge verwendet und deshalb keine Sprungadressen geändert werden müssen.



④ **Flußdiagramm:** Um Töne im Nf-Bereich zu erzeugen, muß das Maschinenprogramm mit zwei Schleifen gebremst werden

Nach dem Eingeben von NEW ist der Computer zur Aufnahme eines Basic-Programms bereit. Im Rahmen dieses Programms kann die Tonmarke einfach mit LET A = USR 23500 aufgerufen werden.

Soll die Tonmarke im SLOW-Modus verwendet werden, empfiehlt es sich, die Werte für die Tonhöhe zu ändern (Inhalt der Speicherstellen 23503 und 23510: jeweils 50). Zu beachten ist, daß jetzt beim Abspeichern des Programms auf Band das Programm zur Tonerzeugung nicht mit abgespeichert wird, da der ZX 81 den oberhalb von RAMTOP liegenden Speicherbereich hierbei nicht berücksichtigt. Das Tonprogramm muß also jeweils vor dem Basic-Programm in der beschriebenen Weise geladen werden.



⑤ **Signale am Datenausgang:** Auch der OUT-Port (Bit 0 bis Bit 7) der Grundplatine kann zum Tonausgang werden. Dabei hat Bit 0 des Datenworts immer die höchste Frequenz

Aktivierte Leitung	Bitmuster	Hex	Dez
A15	0111 1111	7F	127
A14	1011 1111	BF	191
A13	1101 1111	DF	223
A12	1110 1111	EF	239
A11	1111 0111	F7	247
A10	1111 1011	FB	251
A9	1111 1101	FD	253
A8	1111 1110	FE	254

Aktivierte Leitung	Bitmuster	Hex	Dez
keine	0111 1111	7F	127 (31)
KBD 0	0111 1110	7E	126 (30)
KBD 1	0111 1101	7D	125 (29)
KBD 2	0111 1011	7B	123 (27)
KBD 3	0111 0111	77	119 (23)
KBD 4	0110 1111	6F	111 (15)

⑥ **Tastaturabfrage:** Oben die ausgegebenen, unten die eingelesenen Bitmuster*). Bei der Ausgabe ist das Bitmuster spaltenförmig maßgebend (A8...A15)

*) Bild 6 unten. Die ersten drei Bits der Bitmuster werden vom Sinclair-Logik-Chip bestimmt und können unter Umständen nicht die Werte 0 1 1 haben. Bei Maschinenprogrammen hilft dann das Maskieren dieser Bits. So liefert der Befehl *and 31* den reinen Tastencode. Die zugehörigen Zahlenwerte sind im Bild in Klammern gesetzt.

Durch Setzen/Rücksetzen einzelner Bits des Out-Ports (Adresse 23558, siehe Heft 13/83) läßt sich ebenfalls eine tonfrequente Rechteckspannung erzeugen. Besonders elegant gelingt dies durch fortlaufendes Dekrementieren (um 1 verringern) eines CPU-Registers und Ausgeben des Registerinhalts über den Out-Port.

Tonerzeugung über den Out-Port

Bild 5 zeigt, daß an der Bitposition 0 des ausgegebenen Bytes die höchste Frequenz abgenommen werden kann, während die nächsten Bitpositionen eine jeweils um den Faktor 2 niedrigere Frequenz führen. Damit eignet sich diese Methode sehr gut, um mit dem ZX 81 ein Musikinstrument zu realisieren, da aufeinander folgende Oktaven im Frequenzverhältnis 1:2 stehen.

Ein 50-Ω-Lautsprecher bzw. eine 32-Ω-Postkapsel kann über einen Vorwiderstand von 220...470 Ω (Widerstandswert je nach gewünschter Lautstärke) direkt zwischen die entsprechende Bitposition des Out-Ports (DIL 2) und Masse geschaltet werden.

Will man auf der Tastatur des ZX 81 spielen, dann darf der Ton nur erklingen, solange die entsprechende Taste gedrückt ist. Darum muß im Maschinenprogramm zur Tonerzeugung noch eine Tastaturabfrage vorhanden sein.

Wie „liest" der ZX 81 die Tastatur?

Die Tastatur des ZX 81 ist in einer 8×5-Matrix angeordnet (siehe auch Heft 18/1983, Seite 85; bei der dort abgebildeten Tastaturmatrix muß die Verbindung zwischen den Außenkreisen der Tasten V und B schwarz sein, nicht blau!). Dabei sind die acht Zeilen über Dioden mit den Adreßleitungen A8...A15 der CPU direkt verbunden. Die fünf Spalten-(Keyboard)leitungen sind über Widerstände an 5 V gelegt und werden den zugeordneten Eingängen (KBD 0 bis KBD 4) des Sinclair-Logik-Chips zugeführt.

Die Abfrage erfolgt über einen in-Befehl (Maschinensprache). Dabei wird die Tatsache genutzt, daß bei der Ausführung des Befehls *in a,(N)* die untere Adreßbushälfte (A0...A7) durch die CPU mit dem Operanden n belegt wird (der Operand n wählt die Adresse des gewünschten Eingabekanals an), während auf der oberen Adreßbushälfte (A8...A15) der Inhalt des Akkus erscheint.

Ein Beispiel soll das Ganze verdeutlichen: Mit dem Maschinenbefehl *ld a, 127* wird in den Akku das Bitmuster 0111 1111 geladen. Anschließend erfolgt der Befehl *in a,(254)*. Dadurch erscheint auf der Adreßleitung A15 eine logische 0.

Wird nun gleichzeitig eine der A15 zugeordneten Tasten gedrückt, so erhält auch die entsprechende Keyboardleitung (KBD 4...KBD 0) L-Pegel. Das Bitmuster auf den Keyboardleitungen stellt also eine Binärzahl dar, deren Zahlenwert die gedrückte Taste bestimmt. Der Sinclair-Logik-Chip registriert diesen Zahlenwert, führt ihn dem Datenbus zu und lädt den Wert in den Akku.

In *Bild 6* sind die über A8...A15 ausgegebenen Bitmuster (Zeilenauswahl) und die über den Datenbus eingelesenen Bitmuster (Spaltenauswahl) zusammengestellt. Damit kann jede einzelne Taste der Tastaturmatrix angesprochen werden.

Adresse	Mnemonik	Eingabe (dezimal)
16514	nop	0
16515	nop	0
.		0
.		0
16521	nop	0
16522	ld hl,23558	33
16523		6
16524		92
16525	ld b,0	6
16526		0
16527	ld a,(16518)	58
16528		134
16529		64
16530	ld c,a	79
16531	ld a,60	62
16532		60
16533	in a,(254)	219
16534		254
16535	sub 127	214
16536		127
16537	ret z	200
16538	dec c	13
16539	jrnz,−3	32
16540		253
16541	ld (hl),b	112
16542	inc b	4
16543	jr, −18	24
16544		238

⑦ **Orgelprogramm:** Hier ist neben der Tonerzeugung auch gleich die Tastaturabfrage mit enthalten

Jetzt wird der ZX 81 zur Orgel

Kommt der ZX 81 als Musikinstrument zu Ehren, dann sollte man auf der untersten Tastaturreihe die Stammtöne, in der darüberliegenden Reihe die entsprechenden Halbtöne greifen können (Klaviertastatur). Das bedeutet, daß für die Tastaturabfrage die Adreßleitungen A15, A14, A9 und A8 kontrolliert werden müssen. Hierzu gehört das Ausgabe-Bitmuster 0011 1100, d. h., 60 (dezimal) bzw. 3Ch. Falls eine der gewünschten Tasten gedrückt wird, ist der Zahlenwert des eingelesenen Bitmusters ungleich 127. Diese Tatsache nutzt das Maschinenprogramm von *Bild 7*. Eingegeben wird es, wie bereits beschrieben, in eine REM-Zeile ab Adresse 16514.

In Adresse 16518 wird abschließend der für die Tonhöhe maßgebliche Zahlenwert vor Aufruf des Maschinenprogramms eingegeben. Das eigentliche Programm beginnt bei Adresse 16522 (Aufruf: LET A = USR 16522). Ein Tastendruck in der unteren Hälfte der Tastatur erzeugt dann einen Ton.

In den Zeilen 16530...16537 erfolgt die Überprüfung, ob eine der gewünschten Tasten gedrückt ist. Falls dies nicht der Fall ist, bekommt der Akku durch Subtrahieren von 127 den Wert 0, so daß der Rücksprung aus dem Maschinenprogramm erfolgt (ret z).

Für die Tonerzeugung sind zwei Schleifen wichtig: Die Befehle unter den Adressen 16527...16530 laden in das Register c den für die Tonhöhe wichtigen Zahlenwert aus der Speicherstelle 16518. Nachdem der Inhalt von Register c auf 0 heruntergezählt ist (Schleife 1, 16538...16540), wird der Inhalt des b-Registers (zu Programmbeginn 0) über den Port (Adresse 23558) ausgegeben. Anschließend wird das b-Register inkrementiert (um 1 erhöht). Dann wiederholt sich der gesamte Vorgang (Schleife 2, 16527...16544).

Als Hilfestellung und Anregung für eigene Experimente sind in *Bild 8* einige Zahlenwerte (Inhalt der Speicherstelle 16518) und die zugehörigen Frequenzwerte (Tonabnahme an Bit 2 des Ports) angegeben. Besonders hervorgehoben sind die zu den Tönen der zwischen c' und c'' liegenden Oktave (näherungsweise) gehörenden Frequenzen.

Um den ZX 81 als elektronische Miniorgel zu benutzen, muß das Maschinenprogramm in ein kleines Basic-Programm eingebunden werden (*Bild 9*): Das Maschinenprogramm ist wie üblich in einer REM-Zeile (Zeile 100) untergebracht. Wegen Zeile 200 wartet der Computer solange, bis eine Taste gedrückt wird.

Der Variablen A wird der Code-Wert der entsprechenden Taste zugewiesen (Zeile 300) und in Abhängigkeit von diesem Wert die der gewünschten Frequenz entsprechende Zahl in die Speicherstelle 16518 eingeschrieben.

In *Bild 10* ist die „Orgeltastatur" gezeigt. Für einen ersten Probelauf eignet sich der nachfolgende Code, eines sicher allgemein bekannten Liedchens. Ein fetter Buchstabe bedeutet, daß der Ton etwas länger zu halten ist: Z X C V **B B** N N N N **B** N N N N **B** V V V V **C C** X X X X **Z**

Oskar Merker

Stichworte zum Inhalt

Software-Schalter, SR-Flipflop, Tonerzeugung, ZX 81, Maschinenprogramm, akustische Marken, RAMTOP, Tastverhältnis, Tastaturabfrage, elektronische Orgel.

Zahlenwert für Schleife	Frequenz (Hz)	Ton
255	195	
250	199	
200	247	
190	259	
187	264	c'
180	274	
176	280	cis'
170	290	
166	296	d'
160	307	
156	315	dis'
150	327	
149	329	e'
140	350	
139	352	f'
131	373	fis'
130	376	
123	397	g'
120	406	
116	420	gis'
111	438	a'
110	442	
104	467	ais'
100	485	
98	494	h'
91	530	c''

⑧ **Frequenz am OUT-Port:** Die Werte gelten für die Signalabnahme an Bit 2 des Ports und für den FAST-Modus

```
100 REM 0000000000000000000000
        0000000000000000000000
200 IF INKEY$="" THEN GOTO 200
300 LET A=CODE INKEY$
310 LET X=16518
400 IF A=63 THEN POKE X,187
410 IF A=61 THEN POKE X,166
420 IF A=40 THEN POKE X,149
430 IF A=59 THEN POKE X,139
440 IF A=39 THEN POKE X,123
450 IF A=51 THEN POKE X,111
460 IF A=50 THEN POKE X,98
470 IF A=27 THEN POKE X,91
480 IF A=56 THEN POKE X,176
490 IF A=41 THEN POKE X,156
500 IF A=44 THEN POKE X,131
510 IF A=45 THEN POKE X,116
520 IF A=47 THEN POKE X,104
600 LET A=USR 16522
700 GOTO 200
```

⑨ **Frequenzzuweisung:** Dieses Basic-Programm weist den vom Orgelprogramm abgefragten Tasten die gewünschte Frequenz zu

⑩ **Orgeltastatur:** Zwischen der Folientastatur und dem erzeugten Ton besteht der im Schema gezeigte Zusammenhang

ZX 81 à la carte (4):

Schrittmacher

Mit der hier vorgestellten einfachen automatischen Wiederholfunktion (Auto-Repeat) reagieren die „Tasten" des ZX-81-Tableaus auf Dauerdruck.

Auto-Repeat-Schaltungen in Computern sorgen dafür, daß die den Tasten zugeordneten Zeichen oder Funktionen so lange ausgegeben bzw. ausgeführt werden, wie eine Taste gedrückt bleibt. Das bringt offenkundige Vorteile, wenn man z. B. mit dem Cursor eine bestimmte Stelle am Bildschirm erreichen möchte, wenn das Löschen mehrerer Zeichen notwendig ist oder wenn man in einem REM-Kommentar lauter gleiche Zeichen unterbringen will (Platzreservierung für Maschinenprogramm). Die Zahl der Tastenbetätigungen wird dann drastisch reduziert.

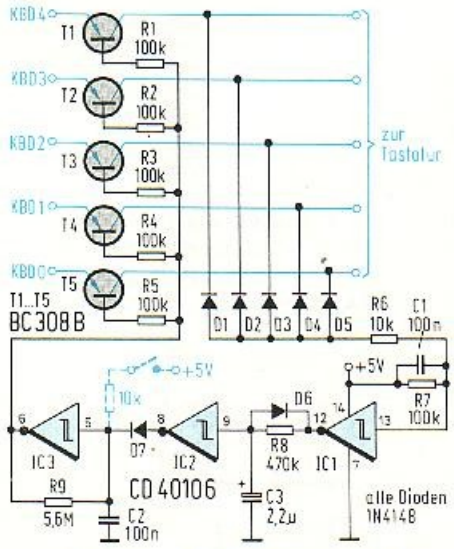
Transistoren kontra Fingertrommeln

Das Schaltbild der Auto-Repeat-Baugruppe zeigt *Bild* 1. Normalerweise sind die fünf von der Tastatur kommenden Leitungen unmittelbar mit den Anschlüssen KBD 0 bis KBD 4 verbunden (Keyboard-Leitungen). Eine Tastenbetätigung hat dann ein charakteristisches Bitmuster auf den Keyboard-Leitungen zur Folge, das vom Sinclair-Logik-Chip ausgewertet wird (siehe auch FUNKSCHAU Heft 18/83 und Heft 24/83).

Aufgabe der Auto-Repeat-Baugruppe ist es, wiederholtes Drücken einer Taste einfach durch rythmisches Unterbrechen der Keyboard-Leitungen zu ersetzen. Die Transistoren T1 bis T5 wirken deshalb als Schalter. Angesteuert werden sie von dem als Oszillator arbeitenden Schmitt-Trigger IC 3. Die Frequenz des Steuersignals hängt von R 9 und C 2 ab; sie ist zugleich die Wiederholrate der Auto-Repeat-Funktion.

Solange keine Taste gedrückt ist (alle Keyboard-Leitungen führen dann H-Pegel), liegt über R 7, IC 1 und IC 2 H-Pegel am Eingang des Oszillators. Der Oszilator kann deswegen nicht schwingen und führt konstant L-Pegel am Ausgang, so daß alle Transistoren leitend sind.

Das ändert sich schlagartig, wenn eine Taste gedrückt wird. Dann liegt auf einer der Keyboard-Leitungen L-Pegel, der über eine der Dioden D 1 bis D 5 auch den Eingang von IC 1 auf L-Pegel zieht. Dadurch führt der Ausgang von IC 1 H-Pegel, und der Kondensator C 3 wird über R 8 langsam aufgeladen. Von der Zeitkonstante dieses Verzögerungsgliedes hängt es ab, mit welcher Zeitverzögerung die Auto-Repeat-Funktion einsetzt. Die Verzögerungszeit ist erforderlich, um überhaupt noch einzelne Buchstaben eingeben zu können. Mit der angegebenen Dimensionierung beträgt die Verzögerungszeit ca. 1 s.

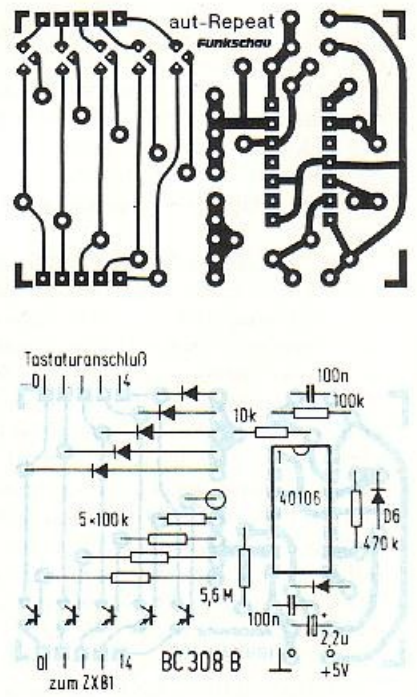


① **Auto-Repeat-Schaltung:** Dieser kleine Zusatz sorgt dafür, daß beim Drücken einer Taste am ZX 81 die jeweils zugeordnete Funktion wiederholt ausgeführt wird

Sobald die Spannung an C 3 die Schaltschwelle des Schmitt-Triggers IC 2 erreicht hat, wird der Kondensator C 2 über D 7 nicht mehr zwangsweise geladen, und der Oszillator IC 3 beginnt zu schwingen, d. h. am Ausgang tritt rhythmisch H-Pegel auf, der die fünf PNP-Transistoren sperrt. Damit ist das Ziel erreicht, die Keyboard-Leitungen kurzzeitig zu unterbrechen.

Nach der Tastenbetätigung sorgt die Diode D 6 für ein Entladen von C 3, damit beim nächsten Tastendruck die Auto-Repeat-Funktion nicht sofort einsetzt. Höhere Werte für R 8 oder C 3 verlängern die Zeitverzögerung, wogegen höhere Werte für R9 und C 2 die Wiederholrate heruntersetzen. Mit der im Schaltbild gestrichelt eingetragenen Ergänzung (auf der Platine *Bild 2* nicht berücksichtigt) läßt sich die Auto-Repeat-Funktion stillegen, wenn z. B. bei Spielen eine softwaremäßige Wiederholfunktion der Cursor-Tasten gegeben ist.

In der gezeigten Ausführung wirkt die Auto-Repeat-Schaltung auf sämtliche Tasten des ZX 81. Wird dagegen nur für die Cursor-Tasten des Computers eine Wiederholfunktion verlangt, so können die Transistoren T 4 und T 5 sowie die

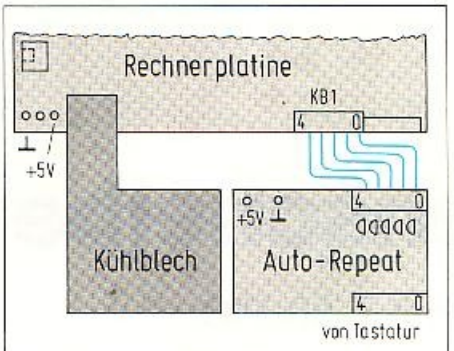


② **Platine und Bestückungsplan:** Die Repeat-Platine paßt im Gehäuse des ZX 81 neben das Kühlblech des Spannungsreglers

zugeordneten Dioden entfallen. Auf der Platine sind dann die betreffenden Kollektor-Emitter-Anschlußpunkte mit Drahtbrücken zu verbinden.

Die Platine paßt ins Rechnergehäuse

Ihren Platz hat die Auto-Repeat-Baugruppe im ZX 81 neben dem Kühlblech des Spannungsreglers. Die Stromversorgung übernimmt der Computer mit seiner +5-V-Betriebsspannung. Um die Repeat-Schaltung an die Rechnerplatine anzuschließen, ist das Versetzen des fünfpoligen Klemmanschlusses für die Folientastatur an die vorgesehene Stelle der Repeat-Platine notwendig. Die dadurch auf der Rechner-Platine freigewordenen Lötstellen werden dann gemäß *Bild 3* mit der Repeat-Platine verbunden. Der Klemmanschluß übernimmt wie bisher das Kontaktieren der Folientastatur.



③ **Montagezeichnung:** Der Klemmanschluß für die Folientastatur (KB 1) muß auf die Repeat-Platine versetzt werden

Den empfindlichen Folienstreifen sollte man erst am Schluß des Einbaus wieder festklemmen, wobei es nicht zu einem Knicken der Folie kommen darf. Am besten packt man den Folienstreifen dazu etwa 1 cm vor seinem Ende mit einer Flachzange oder einer Flachpinzette und schiebt ihn millimeterweise in den Schacht des Klemmanschlusses.

Günter Urbanski

Softwaretip:

Wiederfinden von Programmteilen

Entwickelt man ein langes Programm, so sollte man zum späteren Verständnis wichtige Programmteile mit REM-Kommentaren erklären (strukturieren). Bei den meisten Heimcomputern ist es aber so, daß die Kommentare beim Auflisten wegen einer unübersichtlichen Bildschirmaufteilung schlecht auszumachen sind. Man kann ein Programm aber auch anders strukturieren: Ein Gegenstand soll z. B. über den Bildschirm fliegen und abgeschossen werden. Schriftlich läßt sich dazu folgendes festhalten:

Zeile	Kommentar
100–1000	Variabelendefinition, Start
ab 1000	Bewegung des Gegenstandes
ab 2000	Bewegung des Geschosses
ab 3000	Trefferkontrolle, Punktvergabe
ab 5000	Unterprogramm zur Joystik-Abfrage

So weit, so gut. Korrigiert oder ändert man das Programm, findet man jetzt leicht die Stelle, wo anzusetzen ist. Bei umfangreichen Änderungen bieten jedoch einige Heimcomputer mit RENUMBER die Möglichkeit, zwischen dicht gedrängten Zeilennummern wieder Platz zu schaffen – und danach ist unsere schöne Einteilung wertlos, weil zu den Kommentaren jetzt unbekannte Zeilennummern gehören. Erst ein „Inhaltsverzeichnis" mit zweckentfremdeten GOTO-Befehlen hilft dann weiter:

```
 10 GOTO 100
 20 GOTO 1000: REM BEW. GEGENST.
 30 GOTO 2000: REM BEW.
    GESCHOSS
 40 GOTO 3000: REM TREFFER?
 50 GOTO 5000: REM JOYSTIK
100 A = 3.4: B = 45:...: REM START
```

Da sie wegen Zeile 10 übersprungen werden, haben die GOTO-Befehle in den Zeilen 20 bis 50 auf den Programmablauf keinen Einfluß. Ihren Zweck erfüllen sie erst nach einer Umnumerierung, weil dann auch ihnen die aktuelle Zeilennummer zugewiesen wurde. Nach RENUMBER läßt sich also sofort feststellen, wo die einzelnen Programmteile ihren neuen Platz gefunden haben.

K.-D. Lamann

ZX-81-Softwaretip:

Ladekontrolle durch Prüfsumme

Gelegentlich tritt das Problem auf, daß ein Programm von einer Kassette fehlerhaft (Ändern von Bytes) geladen wird. Ein Prüfsummenprogramm hilft solche Ladefehler – die besonders bei Zahlenwerten folgenschwer sein können – schnell und sicher zu erkennen.

Wenn also ein Basic-Programm zum Speichern fertig ist, sind zuvor noch folgende Zeilen einzugeben:

```
9993 LET A=PEEK 16396+
     256*PEEK 16397-26
9994 FOR B=A TO A+24
9995 INPUT C
9996 POKE B,C
9997 NEXT B
9998 PRINT USR (PEEK 16396+
     256*PEEK 16397-26)
9999 REM 0123456789012345678901 2
     34 (25 Ziffern)
```

Nach dem Start des Hilfsprogramms durch RUN 9993 sind folgende 25 Zahlen einzugeben: 1, 0, 0, 33, 125, 64, 235, 42, 12, 64, 167, 237, 82, 200, 235, 126, 129, 79, 62, 0, 136, 71, 35, 24, 237.

Die Anzeige, die danach erfolgt ist unwichtig.

Nun darf man die Zeilen 9993 bis 9997 wieder löschen. Der Befehl RUN 9998 zeigt schließlich den Wert der Prüfsumme des Basic-Programms an. Notieren Sie diesen (am besten auf der Kassette, auf der das Programm anschließend aufgezeichnet wird). Wann immer das Programm wieder von Kassette geladen wird, läßt sich durch RUN 9998 die Prüfsumme erneut bestimmen und mit der notierten Zahl vergleichen. Stimmen die Zahlen nicht überein, so wurde das Programm falsch geladen.

Die Prüfsumme wird Byte für Byte gebildet vom Programmbereich, d. h. von Adresse 16509 bis zu der Adresse, die von der Systemvariable D-FILE (Bildschirmbereich) angegeben wird, minus Eins. Selbstverständlich beeinflußt jede Änderung des Programms den Wert der Prüfsumme.

Michael Schramm

ZX 81 à la carte (5):

Daten-Drehscheibe

Teil 1: I/O-Schnittstelle mit Z-80-PIO

Der Z-80-PIO-Baustein (PIO: Parallel In Out) verhilft allen Computern mit Z-80-Mikroprozessor zu einer ungemein vielseitigen Ein-/Ausgabe-Schnittstelle.

In den FUNKSCHAU-Heften 12 und 13/1983 wurde für den ZX 81 schon einmal eine I/O-Schnittstelle (I/O-Port) vorgestellt. Deren Platine trägt zusätzlich eine RAM-Erweiterung und ein Flipflop zur Tonerzeugung – sie ist deshalb nicht gerade einfach selbst herzustellen.

Die hier vorgestellte I/O-Schnittstelle bietet mehr Ein-/Ausgabe-Funktionen und beim Aufbau kommt man mit einer einseitig beschichteten Platine aus, die keine Herstellungsprobleme birgt. Auf eine RAM-Erweiterung wurde diesmal verzichtet, da käufliche RAM-Module inzwischen sehr preisgünstig sind. Schnittstelle und RAM-Modul dürfen jederzeit gemeinsam angeschlossen werden, sofern das RAM-Modul das zuläßt (durchgeschleifte Steckerleiste). Die neue Schnittstelle hat jedoch noch einen weiteren Vorteil: Sie läßt sich im Prinzip an jeden Computer mit Z-80-CPU anschließen (ZX 81, ZX Spectrum, VZ 200, Laser 110/210, Jupiter Ace, Colour Genie, MZ 80). Jedoch kann das Betriebssystem des Computers Störungen verursachen, wie Teil 2 am Beispiel des ZX 81 zeigen wird.

Auf die „alte" Schnittstelle konnte man mit den Basic-Befehlen POKE (Daten ausgeben) und PEEK (Daten einlesen) zugreifen. Der PIO-Baustein fordert dagegen Z-80-Maschinenbefehle, die ihn in den gewünschten Betriebszustand (Ein-/Ausgabe) bringen. Die PIO-Schnittstelle muß also programmiert werden.

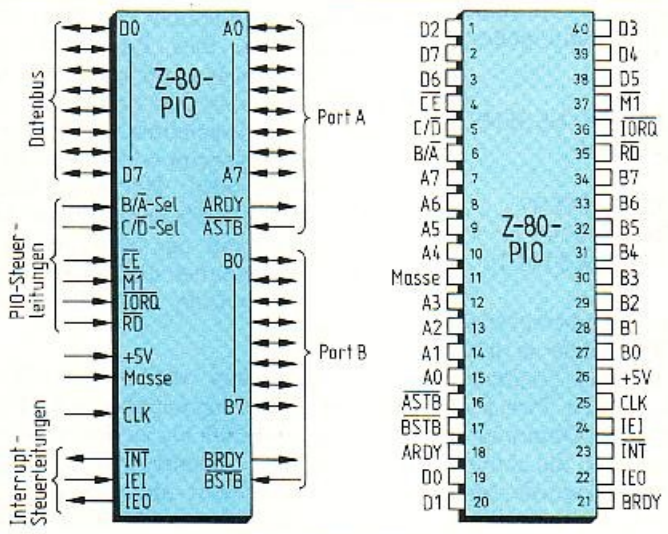
Im dritten Teil werden dafür vielseitige Ein-/Ausgabe-Programme vorgestellt, so daß man die Schnittstelle auch ohne Kenntniss der Z-80-Maschinensprache nutzen kann. Nutzen bedeutet das Steuern externer Geräte oder Anlagen (Ein-/Ausschalten, Lichtorgel, Modelleisenbahn), das Überwachen externer Abläufe oder die Übernahme von Daten (z. B. Eingabe von Morsezeichen).

Z-80-PIO: Tausendsassa im 40poligen Gehäuse

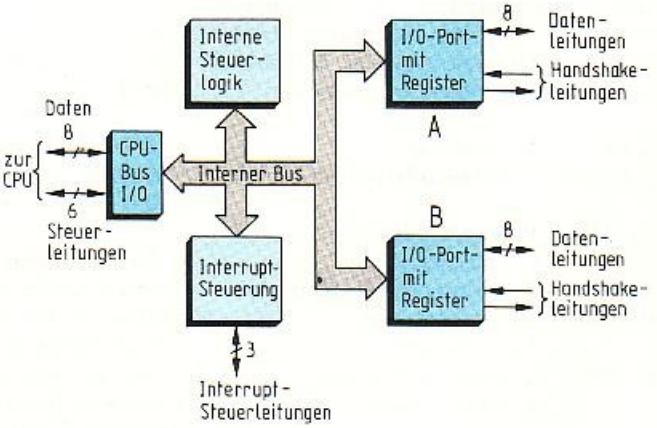
Der PIO-Baustein (*Bild 1*) kann über seine zwei Kanäle (Ports) vom Anwender ausgewählte 8-Bit-Datenworte ausgeben oder an den Ports anliegende 8-Bit-Datenworte zur weiteren Verarbeitung in den Computer holen. Welche der Betriebsarten und welcher Port genutzt wird, muß zuvor mit einem kurzen Maschinenprogramm festgelegt werden.

Darüber hinaus bietet der Baustein eine „Interrupt-($\overline{INT}$-)Steuerung" des Computers. Das heißt, daß der Computer das laufende Programm unterbricht, wenn an einem der Ports ein bestimmter Zustand herrscht (z. B. wenn ein Bit seinen Wert wechselt). Damit wird die CPU entlastet, denn sie muß sich erst dann um den Port kümmern, wenn dort ein vorgesehener Zustand eintritt (beim ZX 81 leider nicht möglich).

Die Betriebsart-Anweisungen müssen dem PIO-Baustein über den Datenbus mitgeteilt werden. Der Baustein „merkt" sich die jeweiligen Anweisungen, indem er sie in Registern (8-Bit-Speicherstellen) zwischenspeichert. Beim Datenver-



① **Z-80-PIO-Baustein:** Das IC enthält sämtliche Bauelemente für eine Parallel-Schnittstelle mit zwei 8-Bit-Ports



② **Blockschaltung des PIO-Bausteines:** Nach Kommandos der CPU an die interne Steuerlogik lenkt diese den Datenfluß selbsttätig

kehr über die Ports greift eine interne Steuerlogik auf diese Register zu und erkennt am Inhalt, wie mit den Daten zu verfahren ist (*Bild 2*).

Der PIO-Baustein enthält bereits alle für die Schnittstelle notwendigen Bauelemente. Damit die CPU das IC gezielt ansprechen kann (adressieren), ist lediglich noch ein Adreßdecoder notwendig (*Bild 3*). Er sorgt dafür, daß der PIO-Baustein immer nur dann auf den Datenbus zugreifen kann, wenn auf dem Adreßbus ganz bestimmte Adressen liegen. Dezimal sind das mit der gezeigten Beschaltung die vier Adressen 248 bis 251 (hexadezimal: F8h, F9h, FAh und FBh). Liegt eine dieser Adressen auf dem Adreßbus, wird der PIO-Baustein über den Freigabeeingang ($\overline{CE}$: Chip Enable) aktiviert.

Zwei Maschinenbefehle lenken den Datenfluß

Da das Betriebssystem das ZX 81 im Adreßbereich zwischen 0 und 8192 untergebracht ist, liegen die Adressen 248 bis 251 eigentlich in einem verbotenen Bereich, denn woher soll der Computer im Ernstfall wissen, ob er die unter diesen Adressen im ROM gespeicherten Befehle ausführen oder den PIO-Baustein aktivieren soll? Mit POKE und PEEK kommt man hier nicht weiter. Erst die Z-80-Maschinenbefehle *in* und *out* helfen aus der Klemme. Dazu ein Beispiel:

Der Befehl *out (F8),a* legt auf den Adreßbus die Adresse F8h und auf den Datenbus den Inhalt des Akkus (gewünschtes Ausgabe-Datenwort). Umgekehrt wird mit *in (F8),a* ein auf dem Datenbus liegendes Eingabe-Datenwort in den Akku geladen. Zusätzlich bewirken beide Befehle, daß der $\overline{IORQ}$-Ausgang der CPU (IORQ: Input/Output-Request – Ein-/Ausgabe-Aufforderung) auf logisch 0 geht.

Abhängig von diesem Signal sperrt beim ZX 81 der Sinclair-Logik-Chip das ROM, so daß es nicht zu einer Doppeladressierung des ROMs und des PIO-Bausteins kommt. Damit auch das PIO-IC erfährt „was los ist", erhält es ebenfalls das $\overline{IORQ}$-Signal (direkt von der CPU).

Unmittelbar mit der CPU verbunden sind noch die Datenleitungen D0 bis D7, die Taktleitung CLK sowie die Steuerleitungen $\overline{M1}$ und $\overline{RD}$. Über $\overline{M1}$ und $\overline{RD}$ teilt die CPU dem PIO-IC mit, ob es Daten vom Datenbus übernehmen kann oder ob es die Daten an einem der Ports auf den Datenbus legen darf (siehe auch Heft 19/1983, Seite 72). Ein Verbinden der $\overline{INT}$-Anschlüsse von CPU und PIO ist allein in der Betriebsart „Interrupt-Steuerung" nötig.

Zwei Nachzügler machen die Adressierung komplett

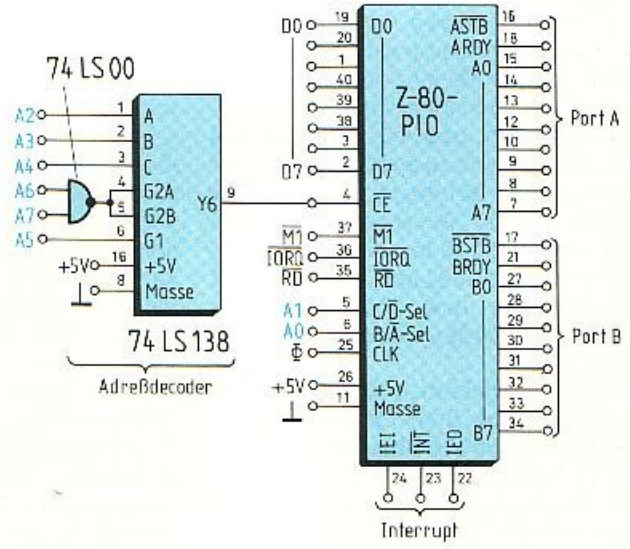
Jetzt fehlen nur noch zwei wichtige Leitungen zur CPU: Die eine hat die Bezeichnung C/$\overline{D}$ SEL und wird mit der Adreßleitung A1 verbunden. Die andere (B/$\overline{A}$ SEL) muß man mit Adreßleitung A0 verbinden.

Erst damit ist die Adressierung des PIO-Bausteins mit den Adreßsignalen A0 bis A7 komplett. Welche der vier PIO-Adressen nun angesprochen wird, entscheidet darüber, ob Port A oder Port B aktiviert wird und ob Daten oder Steuerworte übertragen werden (*Bild 4*). Maßgebend dafür sind die beiden Adreßbits A0 und A1; die restlichen Adreßbits A2 bis A7 dienen lediglich zum Gewinnen des Freigabesignals $\overline{CE}$ über den Adreßdecoder. Hier ließen sich auch andere Adressen als 248 bis 251 wählen, solange man von den 256 möglichen Adressen (0 bis 255) nicht solche verwendet, die auch das Betriebssystem des Computers nutzt.

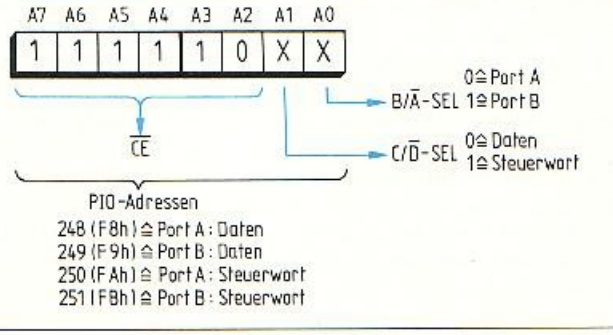
Die Adressierung des PIO-Bausteins ist damit abgeschlossen. Der nächste Schritt ist die Wahl der gewünschten Betriebsart. Dazu muß man dem Baustein unter den Adressen 250 bzw. 251 (FAh bzw. FBh) ein passendes Steuerwort über den Datenbus zuführen. Vier Steuerworte entscheiden dann über vier Betriebsarten.

Vier Betriebsarten erfüllen jeden Wunsch

● In der Betriebsart 0 laßt sich sowohl Port A als auch Port B als Ausgabekanal schalten. Die Daten – jeweils ein Byte – werden dann von der CPU in ein Ausgaberegister des PIO-ICs geschrieben.



③ Komplette I/O-Schnittstelle: Der Adreßdecoder gibt das PIO-IC frei, wenn auf dem Adreßbus Adressen zwischen 248 und 251 liegen



④ Bedeutung der Adreßbits: Von der jeweiligen PIO-Adresse hängt es ab, ob man Daten/Steuerworte für Port A/B auf den Datenbus legen muß

Da nun ein am Port ausgeschlossenes Peripheriegerät nicht genau weiß, wann es die Daten abholen darf, und ebenso die CPU nicht weiß, wann die Daten ausgegeben sind und das Ausgaberegister wieder frei ist, wird der Datenfluß über zwei „Handshake"-Leitungen (Quittungsbetrieb) gesteuert. Jeder Port hat deshalb die Anschlüsse Ready (ARDY, BRDY) und Strobe ($\overline{ASTB}$, $\overline{BSTB}$).

Wenn ein Daten-Byte vollständig im Ausgaberegister ist, zeigt das Ready-Signal durch logisch 1 an, daß von außen auf die Daten zugegriffen werden kann. Wenn dagegen das Peripheriegerät die Daten übernommen hat, meldet es dies dem PIO-Baustein durch einen Strobe-Impuls (logisch 0). Dieser löst im PIO-IC ein Interrupt-Signal aus, das man an die CPU weiterleiten kann. War die CPU inzwischen mit anderen Aufgaben beschäftigt, bringt das Interrupt-Signal sie dazu, das nächste Datenbyte auszugeben.

Der Handshake-Betrieb ist für zeitkritische Programme und bei der Bedienung mehrerer Ein-/Ausgabe-Schnittstellen wichtig, da Peripheriegeräte oft doch wesentlich langsamer arbeiten als die CPU und diese somit übermäßig lange in Anspruch nehmen würden.

Im zeitunkritischen Betrieb darf man die Handshake-Leitungen außer Acht lassen. Auf ein Daten-Byte im Ausgaberegister läßt sich auch ohne Ready-Meldung zugreifen, und der Zeitpunkt des nächsten Datentransports wird ohne Strobe-Impuls (Interrupt) eben von der CPU bzw. vom steuernden Programm bestimmt. Wenn es trotzdem zu Störungen kommen sollte, muß man sich mit Warteschleifen bzw. PAUSE-Befehlen helfen, die allerdings die CPU voll beschäftigen.

● In der Betriebsart 1 kann entweder Port A oder Port B als Eingabekanal geschaltet werden. Ein Eingaberegister auf dem PIO-IC übernimmt dabei die Daten vom Peripheriegerät und die CPU liest sie aus diesem Register.

Auch hier kann man wieder mit den Handshake-Leitungen arbeiten: Ist das Eingaberegister leer – hat also die CPU die Daten übernommen –, wird die Ready-Meldung (logisch 1) ausgegeben. Das Peripheriegerät erkennt das und lädt die nächsten Daten ins Eingaberegister. Ein anschließender Strobe-Impuls kann wieder einen Interrupt bei der CPU bewirken, worauf diese das momentan laufende Programm unterbricht, um die neuen Daten „abzuholen".

Ebenso ist ein Betrieb ohne Handshake-Leitungen erlaubt. Man muß nur beachten, daß die CPU, wenn sie den Befehl zum Auslesen des Eingaberegisters erhält, das liest, was gerade darin steht; egal ob die Übertragung der Daten vom Peripheriegerät abgeschlossen ist oder nicht.

● In der Betriebsart 2 wird Port A für bidirektionalen Datenverkehr – also sowohl für Ein- als auch Ausgabe – benutzt. Port B ist dabei unwirksam und muß auf Betriebsart 3 (siehe unten) eingestellt werden. Zum Steuern bzw. Umschalten zwischen Ein- und Ausgabe werden die Handshake-Leitungen beider Ports verwendet.

Die Ausgabe läuft dann wie in Betriebsart 0 ab, nur muß der Eingang $\overline{ASTB}$ auf logisch 0 sein. Die Eingabe entspricht der in Betriebsart 1 mit der Ausnahme, daß zum Steuern der Dateneingabe über Port A die Handshake-Leitungen von Port B (BRDY und $\overline{BSTB}$) verwendet werden.

● Beide Ports lassen sich in der Betriebsart 3 benutzen. Hierbei kann jedes Bit eines Ports getrennt als Ein- oder Ausgabekanal geschaltet werden. Die Handshake-Leitungen spielen in dieser Betriebsart keine Rolle. Dafür wird eine Interrupt-Möglichkeit geboten: Ein Interrupt wird ausgelöst, wenn sich der Signal-Zustand an einem Eingabekanal ändert, also ein Bit von 1 nach 0 wechselt oder umgekehrt.

Ebenso kann man vorher festlegen, daß erst alle Bits (oder nur bestimmte, durch eine Maske definierte Bits) ihren Wert ändern müssen, bevor ein Interrupt ausgelöst wird. Das Interrupt-Signal kann die CPU selbstverständlich nur dann auswerten, wenn die $\overline{INT}$-Anschlüsse von PIO und CPU verbunden sind.

Dieser Interrupt ermöglicht es, z. B. externe Geräte auf ihren Betriebszustand hin zu überprüfen, ohne dabei die CPU zu belasten. Die Geräte melden sich sozusagen selbst über den PIO-Baustein, wenn z. B. irgend etwas nicht stimmt. Erst dann wird die CPU aktiv, indem sie in die für das entsprechende Gerät (durch einen Interrupt-Vektor) festgelegte Interrupt-Routine springt (mehr darüber später).

Man kann aber auch einen Port in Betriebsart 3 ohne Interrupt-Steuerung benutzen, wenn man z. B. nur ein Bit als Eingabekanal verwenden will; die anderen Bits werden dann als Ausgabekanal geschaltet und erscheinen beim Abfragen des Eingaberegisters immer mit dem logischen Wert 0.

So wird der PIO-Baustein programmiert

Mit Steuerworten unter den Adressen 250 bzw. FAh (für Port A) oder 251 bzw. FBh (für Port B) bringt man das PIO-IC in die gewünschte Betriebsart. Die Steuerworte werden dabei in den Registern des PIO-Bausteins zwischengespeichert, und von der Steuerlogik ausgewertet. In *Bild 5* sind alle zulässigen Steuerworte zusammengefaßt.

Da es außer für die Betriebsart noch andere Steuerworte gibt, wird eine Unterscheidung zwischen ihnen mit Hilfe der niederwertigen vier Bits erreicht. Es gibt aber auch Steuerworte, die unmittelbar auf ein anderes folgen müssen; dabei können alle acht Bits zur Geltung kommen.

Für die Betriebsarten 0, 1 und 2 benötigt man nur das entsprechende Betriebsarten-Steuerwort und, falls Interrupts erlaubt sind, einen Interrupt-Vektor. In diesen Vektor (Register) wird der niederwertige Adreßteil einer Interrupt-Routine geladen, die man bereits im Speicher des Computers abgelegt hat. Der höherwertige Teil wird vorher in das i-Register der CPU geladen.

Bei Auftreten eines Interrupts werden die beiden Adreßteile zusammengefügt und zeigen auf die Anfangsadresse der Interrupt-Routine, bei der das Programm dann fortfährt. Der Rücksprung aus dieser Routine wird mit dem Befehl *reti* erreicht. Danach ist über den PIO-Baustein wieder ein neuer Interrupt möglich.

Dank diesem Interrupt können mehrere PIO-Bausteine parallelgeschaltet sein; abhängig von den Interrupt-Vektoren wird dann bei einem Interrupt die Interrupt-Routine ausgeführt, die dem jeweiligen Baustein zugeordnet ist.

Für die Betriebsart 3 benötigt man neben dem Betriebsarten-Steuerwort und dem Interrupt-Vektor (falls Interrupts erlaubt sind) noch drei weitere Steuerworte:

○ Ein Ein-/Ausgaberegister-Steuerwort, mit dem festgelegt wird, welche Datenleitungen Eingang (logisch 1) und welche Ausgang (logisch 0) sind.

○ Da in der Betriebsart 3 die Handshake-Leitungen unbenutzt bleiben, und Interrupts durch Signaländerungen an den Ports erzeugt werden, ist ein Interrupt-Steuerwort nötig. Es besagt u. a., bei welchem Pegel (0 oder 1) ein Interrupt ausgelöst wird.

○ Sollen einige Bits vom Auslösen eines Interrupts ausgeschlossen werden, so muß man diese in einem folgenden Masken-Steuerwort angeben.
○ Soll ein Interrupt während des Programmablaufes gesperrt oder freigegeben werden, so kann dies mit dem Interrupt-Sperrungswort geschehen. Der Rest des Interrupt-Steuerwortes wird davon nicht verändert.

Die vollständige Programmierung des PIO-Bausteins ist nochmals im Zusammenhang in *Bild 6* dargestellt. Man wählt die gewünschte Betriebsart aus und erkennt, welche Steuerworte dann notwendig sind.

Wenn ein PIO-Baustein nicht genügt

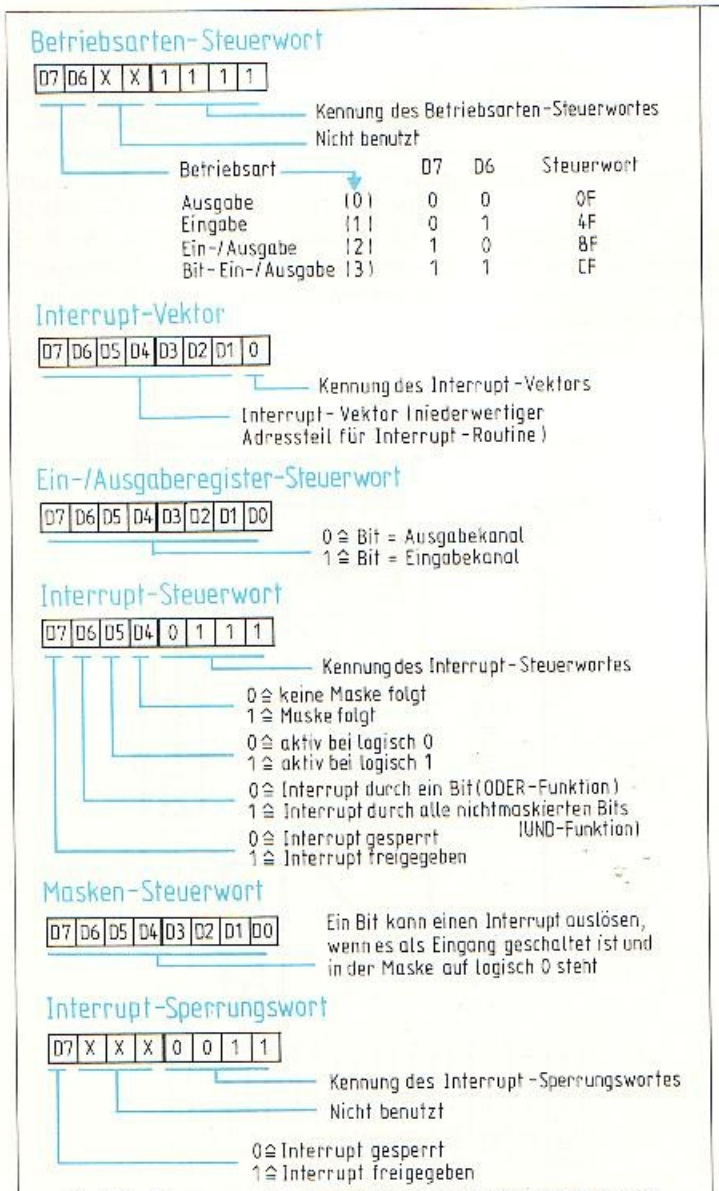
Mit den Signalen IEI (Interruptfreigabe-Eingang) und IEO (Interruptfreigabe-Ausgang) wird ein Interrupt überhaupt erst freigegeben, wenn IEI auf logisch 1 liegt. Beim Verwenden mehrerer PIO-Bausteine wird damit eine Rangfolge (Priorität) der Interrupts erreicht.

Dazu sind die Bausteine in einer „Daisy-chain"-Kette so zusammenzuschalten, daß jeweils ein IEO-Ausgang mit dem IEI-Eingang des nächsten Bausteins verbunden ist. Außerdem muß man alle $\overline{\text{INT}}$-Anschlüsse mit dem der CPU verbinden. Der erste IEI-Eingang liegt fest auf logisch 1 (+5 V). Dieser Baustein hat die höchste Priorität.

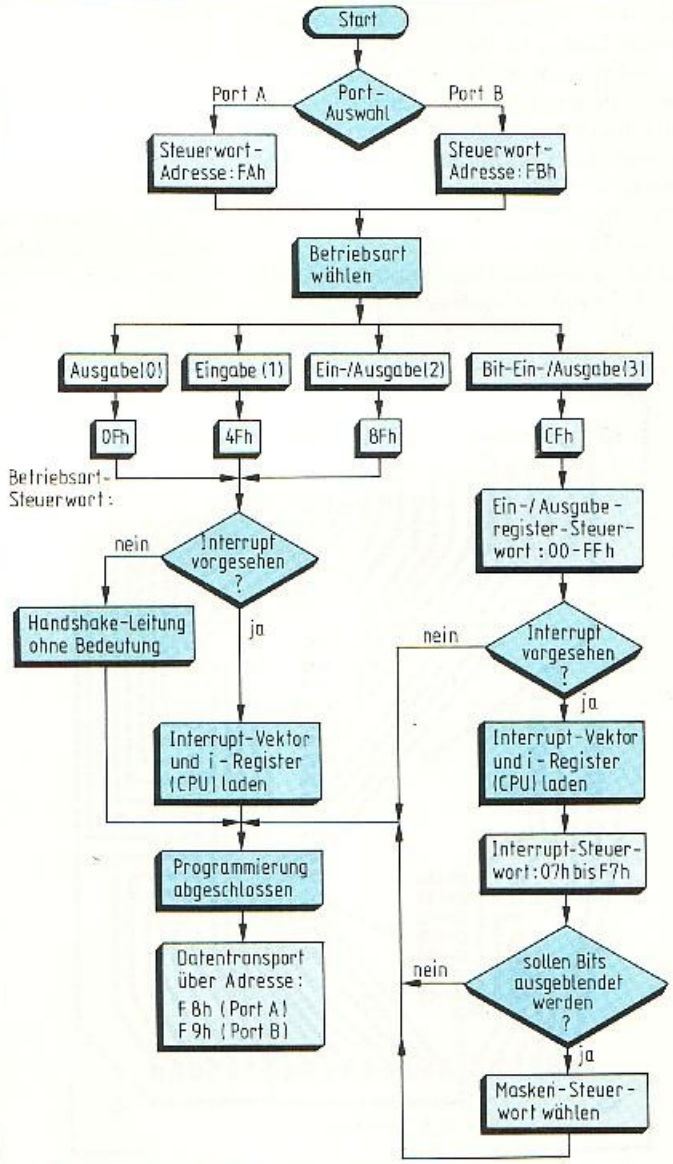
Der Baustein, der einen Interrupt auslöst, unterbricht die Kette und nimmt damit nachfolgenden Bausteinen niedrigerer Priorität die Interrupt-Möglichkeit. Hat man nur einen PIO-Baustein, ist der IEI-Anschluß auf +5 V zu legen, wenn mit der Interrupt-Steuerung gearbeitet werden soll.

Michael Schütz

(Wird fortgesetzt)



⑤ **PIO-Steuerworte:** Immer notwendig ist das Betriebsarten-Steuerwort. Die übrigen Steuerworte sind nur bei bitweiser Ein-/Ausgabe bzw. bei Interrupt von Bedeutung



⑥ **Ermitteln der Steuerworte:** Das Flußdiagramm führt über die Port- und Betriebsart-Anwahl zu den Steuerworten (Wertermittlung im rechten Zweig nach Bild 5)

ZX 81 à la carte (6):

Daten-Drehscheibe

Teil 2: Die komplette Hardware

Beachten Sie den Bestückungsplan: Mehr Bauteile erfordert die Ein-/Ausgabe-Schnittstelle tatsächlich nicht.

Prinzipiell ist die Platine der Ein-/Ausgabe-Schnittstelle (*Bild 1, links*) an jeden Computer mit Z-80-CPU anschließbar, sofern man an die entsprechenden Signalleitungen herankommt (siehe Bild 3 in Teil 1). Außerdem ist darauf zu achten, daß die vier Adressen, unter denen man die Schnittstelle ansprechen kann, nicht auch vom Betriebssystem des Computers verwendet werden (z. B. Bildausgabe, Tastaturabfrage). Ein unkontrolliertes Aktivieren unserer Z-80-PIO wäre die Folge.

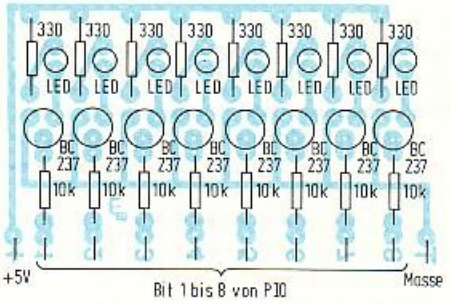
Die im Bestückungsplan des I/O-Ports (*Bild 1, rechts*) eingetragene Beschaltung des Adreßdecoders 74 LS 138 bewirkt, daß die Schnittstelle unter den Adressen F8h, F9h, FAh und FBh anzusprechen ist. Leider werden diese Adressen beim ZX 81 offenbar auch vom Sinclair-Logik-Chip ausgewertet, der außerdem noch die Interrupt-Steuerung verbietet. Dem Betrieb der Schnittstelle am ZX 81 sind daher Einschränkungen auferlegt:

○ Port B darf nur zur Eingabe benutzt werden.
○ Port A erlaubt die Ein- und Ausgabe, sollte aber zur Ausgabe benutzt werden.
○ Die Interrupt-Steuerung ist nicht zulässig; im Handshake-Betrieb kann nur die Ready-Leitung benutzt werden.
○ Der FAST-Modus ist nicht erlaubt.
○ Ist auch der Sinclair-Drucker angeschlossen, kann es bei der Datenausgabe zu einem Papiervorschub kommen.

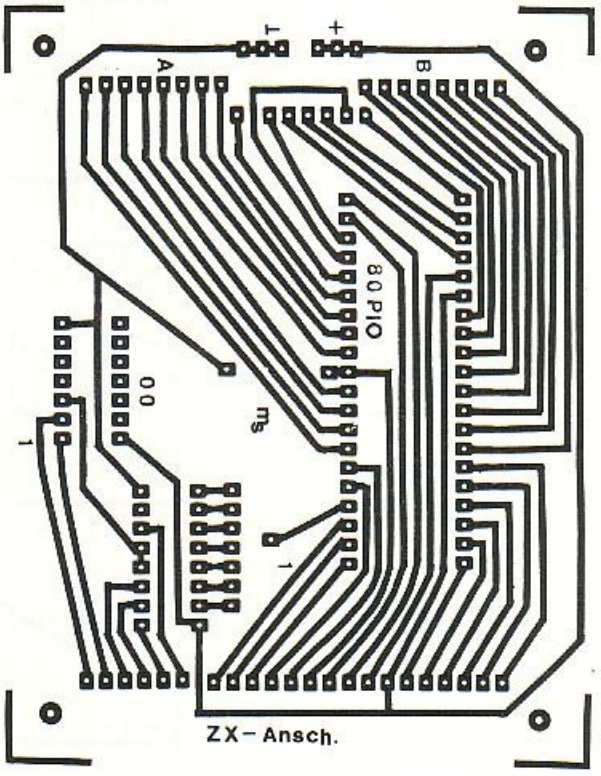
Trotz dieser Einschränkungen läßt sich die Schnittstelle am ZX 81 immerhin noch gut verwenden, wie die Programmbeispiele im nächsten Teil zeigen werden. Nützlich ist auf jeden Fall die in *Bild 2* gezeigte Anzeigeplatine: An Port A angeschlossen (z. B. über Faltstecker) zeigt sie ausgegebene Daten optisch an.

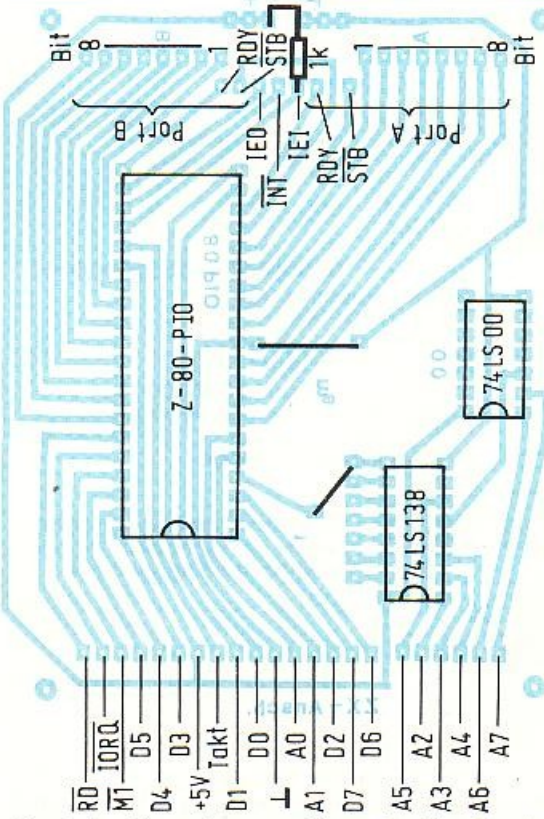
Michael Schütz

(Schluß folgt)



② **Anzeigeplatine:** Da sie so einfach ist, zeigen wir hier nur die Ansicht der Bestükkungsseite (die Ätzvorlage wäre spiegelverkehrt dazu)





① **Ein-/Ausgabe-Schnittstelle:** Die einseitig beschichtete Platine birgt keine Herstellungsprobleme. Rechts: Bestückungsplan und Anschlußanweisung (siehe auch Seite 40/41)

ZX 81 à la carte (7):

Daten-Drehscheibe

Teil 3: Ohne Software geht nichts

Ohne steuernde Software macht die PIO-Schnittstelle keinen Mucks. Mit den hier vorgestellten Demonstrationsprogrammen können wir sie jedoch zum „Leben" erwecken.

Wer sich die PIO-Schnittstelle gemäß Teil 2 nachgebaut hat, der kann die anschließend beschriebenen Ein-/Ausgabe-Programme für den ZX 81 unverändert übernehmen. Die Einschränkungen, die dem Betrieb der Schnittstelle am ZX 81 auferlegt sind (keine Ausgabe über Port B, keine Interrupt-Steuerung, Betrieb nur im SLOW-Modus), werden von den Programmen berücksichtigt.

Der erste Schritt: Hexcode-Eingabe

Wir wissen bereits, daß sich der PIO-Baustein nicht mit Basic-, sondern nur mit Maschinenbefehlen aktivieren läßt (siehe Teil 1). Da der ZX 81 jedoch keinen „Monitor" zum Eintippen von Maschinenprogrammen hat, ist als erstes ein Eingabeprogramm für die Hexadezimalcodes von Maschinenprogrammen notwendig (*Bild 1*).

Das Eingabeprogramm hat die Aufgabe, in Zeile 30 zwischen die Anführungszeichen einzugebende Hexadezimalcodes in Dezimalwerte umzurechnen und diese Werte in die von Nullen reservierten Speicherzellen des REM-Kommentars einzuschreiben. Nach einem Programmstart (RUN) wird deshalb die REM-Zeile ihr Aussehen verändern (siehe auch Heft 12/1983, Seite 76, oder FUNKSCHAU-Sonderheft „Zaubern mit dem ZX 81").

Die dann scheinbar sinnlose Zeichenfolge in der REM-Zeile ist nichts anderes als eine für den ZX 81 lesbare Form des Maschinenprogramms. Um es ausführen zu können, ist der „Aufruf" z. B. mit einer LET-Anweisung, gefolgt von dem Ausdruck USR 16514, nötig.

Die Programmbeispiele zum Steuern des PIO-Bausteins sind dann folgendermaßen aufgebaut: Die erste Programmzeile zeigt das in der Zeichenkette MS untergebrachte Maschinenprogramm (Hexcodes). Diese Zeile ist in unser Eingabeprogramm einzufügen (Zeile 30), das nach dem Start mit RUN den jeweiligen Maschinencode in der REM-Zeile unterbringt.

Anschließend ist das zum jeweiligen Programmbeispiel gehörende Basic-Programm einzutippen. Die Zeilennummern sind dabei so gewählt, daß die Programmzeilen nicht die des Eingabeprogramms überschreiben. Das ist wichtig, denn das Eingabeprogramm ist für jedes Programmbeispiel notwendig. Gestartet mit GOTO 100 übernimmt das Basic-Programm unter anderem den Aufruf des Maschinenprogramms, und die Schnittstelle führt die vom Maschinenprogramm festgelegte Aufgabe (Ein-/Ausgabe) durch.

Der letzte Block in den Programmbeispielen (Adresse, Bytes, Mnemonik) dient lediglich dazu, Kenner der Maschinensprache über die einzelnen Befehlsfolgen der Maschinenprogramme zu informieren. Alles klar? Dann können wir jetzt anfangen, der Ein-/Ausgabe-Schnittstelle „Leben" einzuhauchen.

```
10 REM 00000000000000000000000000
   00000000000000000000000000000
20 LET X=16514
30 LET M$=""
40 FOR I=1 TO LEN M$-1 STEP 2
50 POKE X+INT ((I-1)/2),(CODE
   M$(I)-28)*16+CODE M$(I+1)-28
60 NEXT I
```

① **Eingabeprogramm:** Damit lassen sich die zu den Beispielen gehörenden Maschinenprogramme (Zeile 30) im Hexcode eingeben

```
30 LET M$="3E0FD3FA3E00D3F8C9"

100 INPUT V
110 IF V>255 OR V<0 THEN GOTO100
120 POKE X+5,V
130 LET A=USR X
140 GOTO100

ADRESSE   BYTES   MNEMONIK
4082      3E0F    LD A,0F
4084      D3FA    OUT (FA),A
4086      3E00    LD A,00
4088      D3F8    OUT (F8),A
408A      C9      RET
```

② **Ausgabe eines Datenwortes** zwischen 0 und 255 an Port A. Oben: Maschinenprogramm; Mitte: Basic-Programm; unten: Assembler-Listing des Maschinenprogramms

Zwei Beispiele zur Datenausgabe

Das erste Programmbeispiel (*Bild 2*) zeigt, wie man ein beliebiges Datenwort an Port A der Schnittstelle ausgeben kann. Nach dem Starten des Eingabeprogramms mit der aktuellen Zeile 30 ist das Basic-Programm (Zeile 100 bis 140) einzutippen und mit GOTO 100 zu starten. Der ZX 81 fordert dann zur Eingabe eines Wertes auf (Zeile 100), wobei er nur Werte zwischen 0 und 255 akzeptiert (Zeile 110). Der Wert wird dann in das Maschinenprogramm an die dort durch zwei Nullen reservierte Speicherstelle eingeschrieben (Zeile 120) und anschließend wird dieser Wert mit dem Aufruf des Maschinenprogramms (Zeile 130) an Port A binär ausgegeben. Die in Teil 2 vorgeschlagene Anzeigeplatine macht das deutlich sichtbar.

Nun zum Maschinenprogramm: Zuerst muß es die Betriebsart des PIO-ICs festlegen. Dazu wird der Akku mit 0Fh geladen (Adresse 4082h), dem Steuerwort für die Betriebsart 0 (Ausgabe). Da wir Port A ansprechen müssen, ist anschließend der Akkuinhalt unter der Adresse FAh auszugeben (4084h). Damit ist Port A als Ausgabekanal geschaltet.

Jetzt wird der Akku mit dem auszugebenden Wert geladen (Speicherplatz-Reservierung). Anstelle des Wertes 00, der hier nur Platzhalter ist, schreibt das Basic-Programm mittels POKE den jeweils aktuellen Ausgabewert in das Datenbyte nach dem Ladebefehl.

Die Ausgabe des Akkuinhalts übernimmt wieder ein *out*-Befehl (4088h), wobei die Adresse F8h festlegt, daß Port A angesprochen wird. Der abschließende *ret*-Befehl veranlaßt den Rücksprung ins Basic-Programm, wo Zeile 140 den Sprung zu einer erneuten Dateneingabe einleitet.

Wichtig ist, daß das Basic-Programm nicht mit RUN 100 gestartet wird, weil sonst die Variable X den ihr zugewiesenen Wert 16514 verliert. Wer auf Nummer Sicher gehen will, kann diesen Wert selbstverständlich gleich anstelle von X eintippen.

Ein weiteres Programm zur Datenausgabe zeigt *Bild 3*. Das Maschinenprogramm von Bild 2 ist auch hier unverändert lauffähig. Doch das neue Basic-Programm bewirkt, daß an jedem der acht Ausgänge von Port A ein Schaltimpuls auszugeben ist (z. B. zur Relaissteuerung einer Lauflichtkette), wobei die Einschaltdauer mit dem Wert 50 in Zeile 150 rund 1 s ist. Ersetzt man Zeile 100 durch

100 LET B=INT (RND*9)

dann leuchtet bei angeschlossener Anzeigeplatine immer eine der LEDs nach einem Zufallsmuster kurz auf.

Dateneingabe fordert Pull-up-Widerstände

Das nächste Programmbeispiel erklärt die Dateneingabe über Port B, wobei der Basic-Teil die Datenworte in einem Feld abspeichert (*Bild 4*). Nach zehn Eingaben werden die Werte des Feldes am Bildschirm ausgegeben. Wegen der INPUT-Anweisung in Zeile 120 ist zum Übernehmen des am Port angelegten Datenwortes jedesmal Taste NEWLINE zu drücken.

Zum versuchsweisen Eingeben von Daten sind alle acht Datenleitungen von Port B über „Pull-up"-Widerstände (8 × 4,7 kΩ) an +5 V zu legen. Alle Datenleitungen führen dann H-Pegel (logisch 1). Das entspräche einem Datenwort mit dem Wert FFh (dezimal: 255). Andere Datenworte erhält man, wenn jetzt eine oder mehrere Datenleitungen (z. B. über einen DIL-Schalter) auf Masse gelegt werden. Das zieht an den betroffenen Leitungen die Spannung auf L-Pegel herunter (logisch 0).

Wird z. B. Datenleitung 8 auf L-Pegel gezogen, registriert das Programm den Wert 7Fh (dezimal: 127). Die Daten könnten auch von einem 8-Bit-A/D-Wandler stammen, wenn man z. B. Spannungen messen möchte.

Das Maschinenprogramm legt wieder als erstes durch Ausgabe des Steuerwortes 4Fh unter der Adresse FBh die Betriebsart des PIO-ICs fest: Eingabe an Port B. Anschließend löscht es die beiden Register b und c (laden mit 00), denn diese Register knüpfen die Verbindung zum Basic-Programm. Wird nämlich der *ret*-Befehl erreicht, erhält die Variable im Aufruf des Maschinenprogramms (hier V(K) in Zeile 130) den Inhalt des bc-Registerpaares automatisch zugewiesen. Damit lassen sich Daten einfach aus einem Maschinenprogramm in ein Basic-Programm übernehmen und dort weiterverarbeiten.

Nach dem Löschen des bc-Registerpaares holt der *in*-Befehl das Datenwort an Port B (festgelegt durch die Adresse F9h) in den Akku. Vor dem Rücksprung wird dann nur noch das c-Register mit dem Akkuinhalt geladen. Da stets 8-Bit-Daten anfallen, spielt das b-Register hier keine Rolle. War der Akkuinhalt z. B. 127, so ist genau dieser Wert auch im gerade gültigen Feldelement von V(K) gespeichert.

Die Fähigkeit der Schnittstelle, einen Port als Eingang, den anderen als Ausgang zu benutzen, verdeutlicht das letzte Programmbeispiel (*Bild 5*). Port A ist hier als Ausgang eines ständig von 0 bis 255 „hochlaufenden" Zählers geschaltet.

Über die als Eingang geschaltete Datenleitung 8 von Port B läßt sich der Zähler stoppen, und zwar so lange, wie Datenleitung 8 auf logisch 0 liegt. Diese Datenleitung ist also zunächst wieder über einen Pull-up-Widerstand mit H-Pegel zu versorgen. Ein Taster kann dann die Verbindung mit Masse herbeiführen (L-Pegel) und den Zähler anhalten. Damit steht ein Zahlengenerator bzw. ein Start-/Stop-Zähler (für Meßzwecke) zur Verfügung.

Port B steuert Port A

Als Zähler dient die Variable N des Basic-Programms. Sie wird in Zeile 160 so lange um 1 erhöht und an Port A binär ausgegeben, bis Datenleitung 8 von Port B L-Pegel führt (Meldung STOP am Bildschirm). Solange die Taste gedrückt bleibt, behält auch Port A den zuletzt ausgegebenen Zählerstand bei. Mit dem Loslassen der Taste läuft der Zähler wei-

```
30 LET M$="3E0FD3FA3E00D3F8C9"

100 INPUT B
110 IF B>8 OR B<1 THEN GOTO100
120 POKE X+5,2**(B-1)
130 LET A=USR X
140 POKE X+5,0
150 PAUSE 50
160 LET A=USR X
170 GOTO100
```

③ **Ausgabe eines Schaltimpulses** an einer der Datenleitungen von Port A

```
30 LET M$="3E4FD3FB06000E00
   DBF94FC9"

100 DIM V(10)
110 FOR K=1 TO 10
120 INPUT A$
130 LET V(K)=USR X
140 NEXT K
150 FOR L=1 TO 10
160 PRINT V(L)
170 NEXT L
ADRESSE  BYTES  MNEMONIK
4082     3E4F   LD A,4F
4084     D3FB   OUT (FB),A
4086     0600   LD B,00
4088     0E00   LD C,00
408A     DBF9   IN (F9),A
408C     4F     LD C,A
408D     C9     RET
```

④ **Eingabe von Datenworten** über Port B. Nach der zehnten Eingabe werden alle Datenworte am Bildschirm angezeigt

```
30 LET M$="3E0FD3FA3E00D3F8
   C93ECFD3FB3E80D3FB06000E
   00DBF94FC9"

100 LET N=0
110 POKE X+5,N
120 LET A=USR X
130 PRINT AT 0,0;"START"
140 LET B=USR (X+9)
150 IF B=0 THEN GOTO190
160 LET N=N+1
170 IF N>255 THEN LET N=0
180 GOTO110
190 PRINT AT 0,0;"STOP "
200 GOTO140
ADRESSE  BYTES  MNEMONIK
4082     3E0F   LD A,0F
4084     D3FA   OUT (FA),A
4086     3E00   LD A,00
4088     D3F8   OUT (F8),A
408A     C9     RET
408B     3ECF   LD A,CF
408D     D3FB   OUT (FB),A
408F     3E80   LD A,80
4091     D3FB   OUT (FB),A
4093     0600   LD B,00
4095     0E00   LD C,00
4097     DBF9   IN (F9),A
4099     4F     LD C,A
409A     C9     RET
```

⑤ **Kombinierte Ein-/Ausgabe:** Port A ist der Ausgang eines Zählers. Ein Eingangssignal an Port B kann diesen Zähler stoppen

ter. Am Bildschirm wird das mit der Meldung START quittiert.

Das Maschinenprogramm besteht diesmal aus zwei Teilprogrammen. Das erste ist das Ausgabeprogramm von Bild 2. Es übernimmt die Ausgabe der Zähler-Variablen N und wird in Zeile 120 aufgerufen. Der erste ret-Befehl (Adresse 408Ah) bewirkt dann den Rücksprung in das Basic-Programm.

Neun Bytes nach dem ersten Teilprogramm beginnt das zweite. Der Aufruf in Zeile 140 lautet deshalb

LET B=USR (X+9)

Mit dem unter Adresse FBh erteilten Steuerwort CFh wird Port B auf Betriebsart 3 eingestellt (Bit-Ein-/Ausgabe). Gemäß Teil 1 ist darüber hinaus noch ein Ein-/Ausgaberegister-Steuerwort erforderlich. Wählen wir dafür den Wert 80h (Ausgabe unter Adresse FBh), dann ist Datenleitung 8 von Port B als Eingang geschaltet. Die übrigen Datenleitungen ermöglichen keine Eingabe, sie sind als Ausgang geschaltet.

Der Rest des Maschinenprogramms stimmt mit dem Eingabeprogramm von Bild 4 überein. Nach dem Rücksprung ins Basic-Programm meldet die Variable B, ob die Taste gedrückt oder nicht gedrückt war.

Bei allen Programmbeispielen wurde das jeweilige Maschinenprogramm mit jedem Aufruf vollständig abgearbeitet und somit die Steuerworte jedesmal neu ausgegeben. Das PIO-IC speichert jedoch die Steuerworte in Registern. Deshalb ist es auch zulässig, bei einem wiederholten Aufruf der Maschinenprogramme die Definition der Betriebsart zu überspringen.

Michael Schütz/-ll

ZX-81-Softwaretip:

Simulieren der Fixkomma-Darstellung

Bei vielen Basic-Computern erhält man eine formatierte Ausgabe von Zahlen durch Anwendung des USING-Befehls (Fixkomma-Ausgabe). Damit lassen sich Tabellen sehr übersichtlich gestalten, da bereits vor dem eigentlichen Ausdruck die Vor- und Nachkommastellen einer Zahl wunschgemäß festgelegt werden.

Mit einem kurzen Programm (*Bild 1*) können auch ZX-81-Anwender die Vorzüge der Fixkommaausgabe genießen. Damit ist es möglich, verschiedene Zahlen-Formate durch Unterprogrammaufruf darzustellen. Die Anlehnung an die übliche USING-Befehlsfolge erleichtert den Umgang mit dem Programm.

Das USING-Programm besteht aus zwei Unterprogrammen: Im ersten (ab Zeile 9000) wird das auszugebende Format mittels des Strings U$ definiert. Im zweiten (ab Zeile 9200) wird die Variable Y, die im Hauptprogramm den unformatierten Zahlenwert repräsentiert, formatiert ausgegeben.

Bei normalen USING dienen die Zeichen Doppelkreuz und Dezimalpunkt dem Festlegen des Formats. Da im ZX-81-Zeichensatz das Doppelkreuz nicht enthalten ist, wird es durch „$" ersetzt. Die Folge der Dollarzeichen legt die Anzahl der jeweiligen Stellen fest, ein Punkt die Position des Dezimalpunkts. Dabei ist zu beachten, daß für ein mögliches Vorzeichen eine Vorkommastelle mehr gesetzt werden muß. Um beispielsweise ein Format von fünf Vor- und drei Nachkommastellen zu definieren, muß der Aufruf des ersten Unterprogramms im Hauptprogramm lauten:

LET U$="$$$$$.$$$"
GOSUB 9000

Anschließend kann mit „GOSUB 9200" die Ausgaberoutine aufgerufen werden. Die Variable Y wird dann, wurde ihr zuvor im Hauptprogramm ein Wert zugewiesen, mit der angegebenen Stellenzahl auf dem Bildschirm erscheinen (*Bild 2*). Ist eine Darstellung im gewünschten Format nicht möglich, so erhält man als Fehlermeldung ein „E". Besteht U$ nur aus einer Folge von Dollarzeichen (ohne Punkt), so wird der Integeranteil von Y ausgegeben; vorausgesetzt, eine ausreichende Stellenzahl war definiert.

Der Betrag der Variablen Y darf zwischen 9.9999999E12 und 1E-31 liegen oder gleich Null sein. Bei zu großen Werten ergibt sich ein nicht korrekter Ausdruck, bei zu kleinen erhält man die Fehlermeldung 6/9390.

Für USING üblich ist die nicht durchgeführte Rundung der letzten Stelle. Bei Zahlen mit mehr als sechs Nachkommastellen kann es außerdem durch die Umrechnung ab Zeile 9390 vorkommen, daß die letzte gültige Stelle der Ausgabe fälschlicherweise „aufgerundet" wird. Da der Fehler jedoch nur die Darstellung betrifft, der Wert der Variablen Y aber unverändert bleibt und für weitere Berechnungen zur Verfügung steht, ist diese Ungenauigkeit zu vernachlässigen.

Wolfgang Schaaf

```
9000 REM USING-DEFINITION
9010 REM U$ BESTIMMT FORMAT
9020 REM Z.B. U$="$$$$.$$$"
9030 LET V=0
9040 FOR I=1 TO LEN U$
9050 IF U$(I)="." THEN GOTO 9110
9060 LET U$(I)=" "
9070 IF V<>0 THEN LET U$(I)="0"
9080 NEXT I
9090 IF V=0 THEN GOTO 9110
9100 RETURN
9110 LET V=I-1
9120 LET U$(V)="0"
9130 GOTO 9080
9200 REM USING AUSGABE
9210 REM Y WIRD FORMATIERT
9230 LET N$=" "
9240 IF Y<0 THEN LET N$="-"
9250 LET Z=ABS Y
9255 IF Z=0 THEN LET Z=1E-31
9260 LET L=INT (LN Z/LN 10)
9265 IF L<-5 THEN GOTO 9390
9270 IF L<-1 THEN LET N$=N$+"0"
9280 IF L<0 THEN LET L=0
9290 IF L>=V-1 THEN GOTO 9370
9300 LET V$=N$+STR$ Z
9310 LET W$=U$
9320 LET X=V-L-2+LEN V$
9330 IF X>LEN W$ THEN LET X=LEN
W$
9340 LET W$(V-L-1 TO X)=V$
9350 PRINT W$
9360 RETURN
9370 PRINT "E"
9380 RETURN
9390 LET Z=Z*10**(7-L)
9400 LET V$="00"+STR$ Z
9410 LET W$=U$
9420 LET W$(V-1)=N$
9430 GOTO 9320
```

◄ ① **Listing „USING":** Das Programm formatiert erst eine Zahl und gibt sie dann aus

```
331.45142          331.451
859.61914          859.619
56.452026           56.452
984.97009          984.970
74.493408           74.493
588.05847          588.058
40.512085           40.512
39.489746           39.489
962.82959          962.829

10 LET Y=RND*1000
20 PRINT Y/
30 LET U$="$$$$$.$$$"
40 GOSUB 9000
50 GOSUB 9200
60 GOTO 10
```

② **Demonstrationsbeispiel:** Formatiert (rechts) läßt sich die Zahlenkolonne eindeutig besser lesen. Unten das zugehörige Hauptprogramm

Nachtrag:

Entfesselte PIO

Die ab Seite 18 in der Beitragsfolge „Daten-Drehscheibe" vorgestellte PIO-Schnittstelle hat jetzt für den ZX 81 ihre „Schrecken" verloren. Es genügt, die Verdrahtung zu ändern. Damit werden fast alle Einschränkungen aufgehoben.

Bislang war die „Daten-Drehscheibe" für den ZX 81 tatsächlich noch nicht ganz rund, gab es doch beim Betrieb der PIO (Ein-/Ausgabe-Schnittstelle) einige empfindliche Einschränkungen: Port B durfte nur zur Dateneingabe benutzt werden, der FAST-Modus war verboten, und wurde außer der PIO auch ein Sinclair-Drucker angeschlossen, dann machte dieser zuweilen, was er wollte.

Schon durch eine geänderte Adressierung der PIO fallen alle diese Einschränkungen weg. Nur eine bleibt: Die Interrupt-Betriebsart ist nach wie vor unzulässig.

Störquelle ist der Logik-Chip

Die Schuld an den Problemen, die der ZX 81 mit der nomalerweise für alle Z-80-Computer geeigneten PIO-Schnittstelle hat, ist wieder einmal die unvollständige Adreßdecodierung im ZX 81. Diesmal ist es aber nicht – wie in Heft 10 beschrieben – das ROM, das den Ärger macht, sondern der Sinclair-Computer-Logik-Chip (SCL).

Betrachten wir dazu die Adressierung der PIO (*Bild 1*). Mit L-Pegel am Ausgang $\overline{\text{IORQ}}$ (input/output-request) macht die Z-80-CPU den PIO-Baustein darauf aufmerksam, daß die auf den Adreßleitungen A0 bis A7 liegende Adresse ihn betrifft und nicht etwa ein Speicher-IC.

Wie auf Seite 19 nachzulesen ist, muß sich die PIO unter vier Adressen ansprechen lassen, um Steuerworte oder Daten für Port A oder Port B übertragen zu können. Werden dafür vier aufeinanderfolgende Adressen ausgewählt, z. B. 0, 1, 2, 3 oder 4, 5, 6, 7, dann läßt sich die PIO direkt mit den Signalen A0 und A1 adresssieren.

Wenn jetzt der Adreßdecoder D1 noch dafür sorgt, daß die PIO nur bei einer einzigen dieser Adreß-Vierergruppen aktiv wird, dann haben wir eine vollständige Adreßdecodierung. Das heißt, eine einzige Bitkombination auf den Leitungen A2 bis A7 gibt die PIO über den $\overline{\text{CE}}$-Eingang (Chip-Enable) frei; alle übrigen Adressen stehen für andere Ein-/Ausgabe-Bausteine zur Verfügung.

Zu einer unvollständigen Adreßdecodierung kommt es, wenn der Adreßdecoder z. B. nur mit den Signalen A2 bis A6 gespeist wird. Dann wird die PIO schon von zwei Adreß-Vierergruppen aktiviert. Die eine Vierergruppe liegt im Adreßbereich 0 bis 127, die andere im Adreßbereich 128 bis 255.

Unbedenklich ist diese unvollständige (billigere) Adreßdecodierung nur, solange sich nicht verschiedene Ein-/Ausgabe-Bausteine deshalb in die Haare geraten. Und genau das ist beim ZX 81 der Fall: Die Firma Sinclair hat nämlich dem SCL-Chip, der auch ein Ein-/Ausgabe-Baustein ist, nicht einmal eine unvollständige, sondern gar keine Adreßdecodierung zugestanden. Deshalb reagiert der SCL-Chip unkontrolliert auf alle möglichen Adressen für Ein-/Ausgabe-Bausteine. Es gibt aber zwei Möglichkeiten, ihn daran zu hindern.

Ein Adreßdecoder bändigt den Logik-Chip

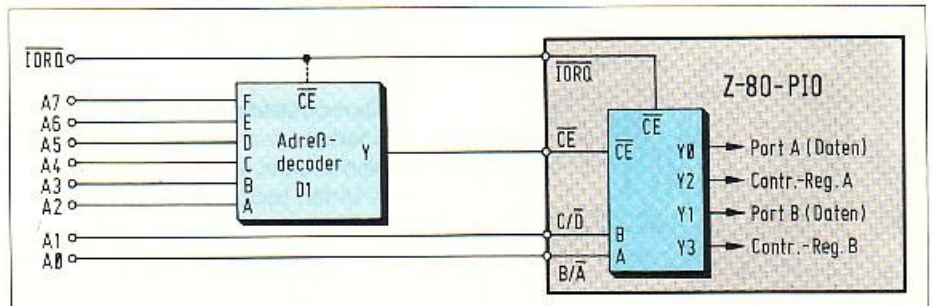
An Pin 7 hat der SCL-Chip einen $\overline{\text{IORQ}}$-Eingang, wie ihn auch die PIO hat. Sobald die CPU auf diesen Eingang L-Pegel legt, wird der SCL-Chip aktiviert und treibt sein Unwesen. Gebändigt ist das IC, wenn es gelingt, nur dann $\overline{\text{IORQ}}$-Signale bis zu Pin 7 durchzulassen, wenn auch tatsächlich der SCL-Chip angesprochen werden soll.

Diese Aufgabe übernimmt die Schaltung in *Bild 2*. Sie nutzt die bei der PIO-Bauanleitung übriggebliebenen drei NAND-Gatter des 74LS00, um den SCL-Chip nachträglich mit einem Adreßdecoder zu versehen.

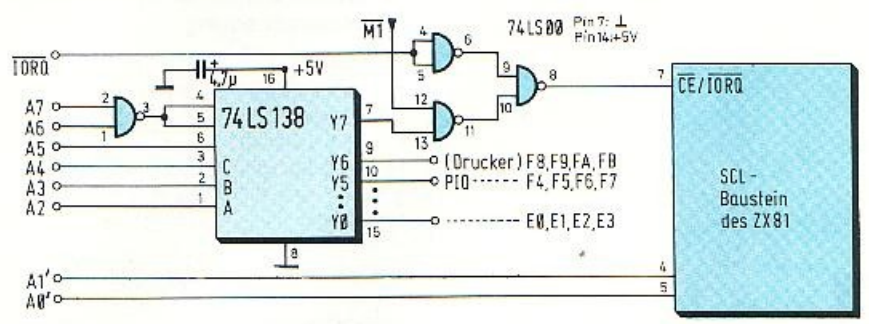
Die drei NAND-Gatter sind zu einem ODER-Gatter geschaltet, das nur dann das $\overline{\text{IORQ}}$-Signal zu Pin 7 durchläßt, wenn Ausgang Y7 des Adreßdecoders L-Pegel führt. Die ursprünglich zu Pin 7 führende Leitung muß deshalb auf der ZX-81-Platine aufgetrennt werden.

Die übrigen Ausgänge des Adreßdecoders können jetzt die Freigabe von bis zu sieben PIOs oder anderer Ein-/Ausgabe-Bausteine übernehmen. Diese haben dann, jeweils in Vierergruppen pro Ausgang zusammengefaßt, die Adressen 224 bis 251 (E0h bis FBh). Für die Adressen 0 bis 223 wären weitere Adreßdecoder erforderlich.

Wegen der in Bild 2 gezeigten Ansteuerung des Adreßdecoders wird der SCL-Chip jetzt nur noch dann angespro-



① **PIO-Adressierung:** Nur ein einziger vier Adressen umfassender Adreßblock ist für die PIO gültig



② **Lösungsvorschlag 1:** Der SCL-Chip im ZX 81 bekommt einen Adreßdecoder vorgesetzt. Das PIO-IC wird z. B. vom Signal Y5 des 74LS138 freigegeben

chen, wenn eine für ihn gültige Adresse vorliegt. Das sind die vom Betriebssystem benötigten Adressen 253, 254, 255 (FDh, FEh, FFh) und, da wieder Viererergruppen decodiert werden, die Adresse 252 (FCh). Alle übrigen Adressen dürfen ohne Komplikationen verwendet werden.

Auch „Spezialadressen" führen zum Ziel

Die PIO läßt sich sogar ohne Eingriff in den ZX 81 „entfesseln". Die ungezügelte Aktivität des SCL-Chips verursacht nämlich keine Probleme, solange bei einer Ein-/Ausgabe die Adreßleitungen A0 und A1 auf H-Pegel liegen. Das veranlaßt den SCL-Chip lediglich, Daten an den UHF-Modulator auszugeben. Außer eventuell einem Flackern auf dem Bildschirm, hat dies auf die Zusammenarbeit zwischen PIO und ZX 81 keine Auswirkungen.

Bei der Schaltung nach *Bild 3* wird von dieser Möglichkeit Gebrauch gemacht. Die PIO (oder sieben andere Ein-/Ausgabe-Bausteine) werden jetzt nur noch dann aktiviert, wenn auch die Bedingung „A0 und A1 auf H-Pegel" erfüllt ist. Gegenüber der ursprünglichen Schaltung auf Seite 19 ist allein die Ansteuerung mit den Signalen A0 bis A7 verändert. Selbstverständlich müssen aber auch in den Steuerprogrammen für die PIO (bei den *in*- und *out*-Befehlen) die neuen Adressen berücksichtigt werden.

Bei der Schaltung nach Bild 3 verbleiben von den 256 möglichen Ein-/Ausgabe-Adressen nur 32 für die PIO und andere Bausteine. Es wurde nämlich gleich noch eine dritte Bedingung erfüllt: Auch A2 muß bei der Adressierung H-Pegel haben, damit nicht ungewollt der ZX-Drucker anfängt zu arbeiten.

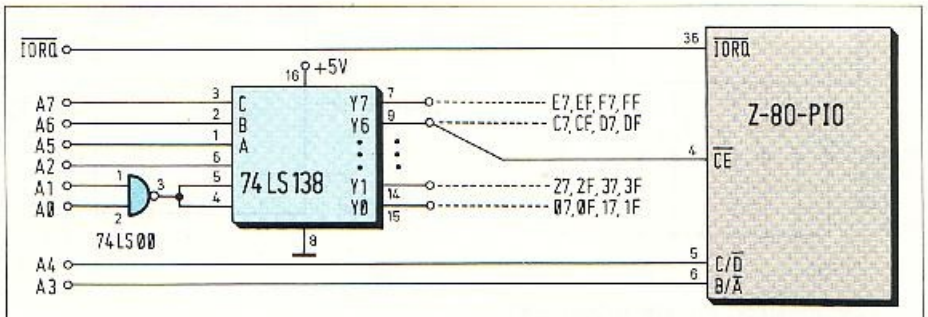
So wird der Drucker gezügelt

Auch der ZX-Drucker ist nur unvollständig decodiert. Er fühlt sich immer dann angesprochen, wenn bei einer Ein-/Ausgabe-Adresse A2 L-Pegel führt. Vom Betriebssystem des ZX 81 wird er über die Adresse 251 aktiviert. Das ist eine unkritische Adresse, bei der A0 und A1 auf H-Pegel liegen, so daß der SCL-Chip friedlich bleibt.

Um den Drucker auch mit der Schaltung nach Bild 2 betreiben zu können, muß er (fast) voll decodiert werden. Dazu ist die Adreßleitung A2 im Anschlußstecker des Druckers abzulöten, und mit dem Ausgang Y6 des Adreßdecoders (Bild 2) zu verbinden. Das Signal A2 führt im ZX-Drucker zum $\overline{CE}$-Eingang eines Steuer-Chips.

Jetzt sind immerhin noch die Adressen 0 bis 247 frei verwendbar. Obwohl der Drucker eigentlich nur eine Adresse benötigt, belegt er nun – da der Adreßdecoder stets Viererergruppen freigibt – die Adressen 248 bis 251 (F8h bis FBh).

Horst Schmid/-ll



③ **Lösungsvorschlag 2:** Gegenüber der ursprünglichen Schaltung sind lediglich die Leitungen A0 bis A7 neu verdrahtet, um kritische Adressen zu vermeiden

ZX-81-Softwaretip:

Zeit sparen bei SAVE und LOAD

Beim Speichern eines Programms auf Kassette werden auch die Variablen mit aufgezeichnet. Oft werden diese jedoch gar nicht benötigt, z. B. dann, wenn das Programm durch RUN gestartet wird, wodurch alle Variablen ohnehin gelöscht werden. In solchen Fällen spart es Zeit, Variablen vor dem Speichern mit CLEAR zu löschen.

Beim Arbeiten mit einem 16-KByte-Zusatzspeicher ist es sehr wirkungsvoll, vor dem Speichern einzugeben: POKE 16389,68.

Hierdurch wird die Systemvariable RAMTOP auf einen Wert gesetzt, der dem des ZX81 ohne Zusatzspeicher entspricht. Das hat zur Folge, daß das Betriebssystem glaubt, es müsse mit dem Speicherplatz besonders sparsam umgehen. Aus dem Bildspeicher werden dann alle Leerstellen gestrichen! Da dieser Speicherbereich mit auf die Kassette gelangt, werden dadurch beim Aufzeichnen wie auch beim Laden eines Programms etwa 17 s Zeit gespart. Vor dem Laden ist RAMTOP durch POKE 16389,128 (oder Unterbrechen der Stromversorgung) wieder auf den ursprünglichen Wert zu bringen.

Michael Schramm

ZX 81 à la carte (8):

Hardware voller Klang

Teil 1: Grundlagen

Die Tonerzeugung per Hardware hat den Vorteil, daß der Computer Zeit zum Erledigen anderer Aufgaben findet.

Die softwareorientierte Tonerzeugung, die ab Seite 12 vorgestellt wird, ist sehr vielseitig, da sämtliche Parameter (Impulsbreite, Frequenz usw.) über ein Programm eingestellt werden können. Andererseits wird der Computer von der Tonerzeugung vollständig in Anspruch genommen, das heißt, er kann währenddessen keine anderen Aufgaben übernehmen. Und nachdem ein Tonprogramm vorzugsweise im FAST-Modus läuft, fehlt schließlich auch die Bildausgabe.

Die hardwareorientierte Tonerzeugung vermeidet alle diese Nachteile. Voraussetzung ist lediglich ein 8-Bit-Parallel-Port, wie er z. B. auch ab Seite 18 vorgestellt wird.

Geschaltete Kondensatoren sind der Schlüssel

Die Tonvariation erfolgt dadurch, daß der Wert eines frequenzbestimmenden Bauteils in einem Tonoszillator vom Computer verändert wird (*Bild 1*). Über acht Datenbusleitungen wären somit zumindest acht verschiedene Frequenzen schaltbar.

Werden jedoch Bauteile verwendet, deren Werte sich bei Parallelschaltung addieren, so kann man durch eine Gewichtung der über die Datenleitungen D0...D7 geschalteten Bauelemente im Verhältnis 1 : 2 : 4 : 8 : 16 : 32 : 64 : 128 erreichen, daß der Gesamtwert proportional zum ausgegebenen Datenwort ist. Diese Bedingung ermöglicht es, 255 verschiedene Töne zu erzeugen; sie wird von Kondensatoren erfüllt.

Falls bei der verwendeten Oszillatorschaltung die Frequenz oder die Schwingungsdauer ebenfalls proportional zum Bauteilewert ist, so besteht letztlich auch Proportionalität zwischen dem Datenwort und der Frequenz bzw. Schwingungsdauer.

Ein Abstecher in die Analogtechnik

Die Oszillatorschaltung stützt sich auf den Timer 555 (*Bild 2*): Der Kondensator C wird zunächst über die Widerstände R_A und R_B aufgeladen. Erreicht die Spannung an Pin 6 einen Schwellenwert, so wird C durch einen internen Entladetransistor (zwischen Pin 7 und Masse) über R_B entladen. Unterschreitet die Spannung an Pin 6 einen tieferen Schwellenwert, dann sperrt der Entladetransistor wieder und der gesamte Vorgang beginnt von neuem.

Die Daten der Schaltung sind in der *Tabelle* zusammengefaßt. Um eine Frequenzvariation zu erreichen, können R_A, R_B oder wie in unserem Fall der Kondensator C umgeschaltet werden (*Bild 3*). Dann gilt:

$C_{ges} = C_0 + C_1 + C_2 + ... + C_7$

$T \sim C_{ges}$; daraus folgt:

$T \sim N$ (N: 8-Bit-Datenwort).

Die Frequenz ist somit umgekehrt proportional zum Wert des Datenwortes. Das bedeutet, daß die Frequenzabstände für kleine Werte von N groß sind (maximale Frequenzänderung 2 : 1), für große Werte von N dagegen klein (minimale Frequenzänderung 255 : 254 = 1,004 : 1).

Vorteilhaft ist, daß bei einer Frequenzumschaltung das Tastverhältnis nicht verändert wird. Wählt man nämlich $R_B \gg R_A$, so wird $T_V \approx 1 : 1$.
Durch entsprechende Dimensionierung von R_A und R_B kann der überstrichene

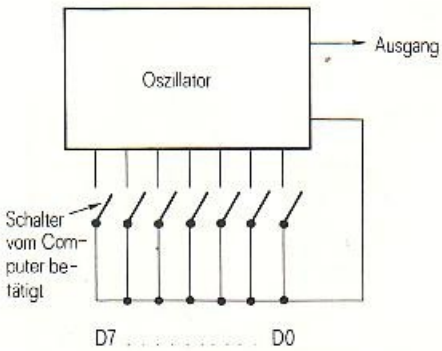
Tabelle: Technische Daten für die Oszillatorschaltung

Aufladezeit:
$t_1 = 0{,}7 \times (R_A + R_B) \times C$
Entladezeit:
$t_2 = 0{,}7 \times R_B \times C$
Periodendauer:
$T = t_1 + t_2 = 0{,}7 \times (R_A + 2 \times R_B) \times C$
Frequenz:

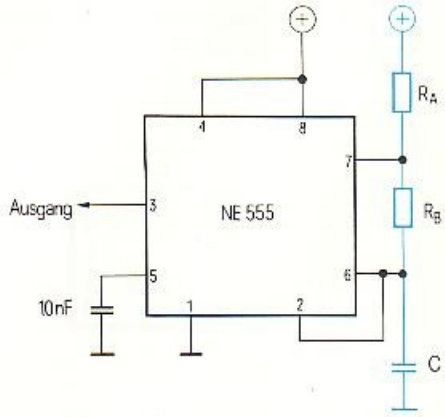
$$f = 1/T = \frac{1{,}44}{(R_A + 2 \times R_B) \times C}$$

Tastverhältnis:

$$T_V = t_2/t_1 = \frac{R_B}{R_A + R_B}$$



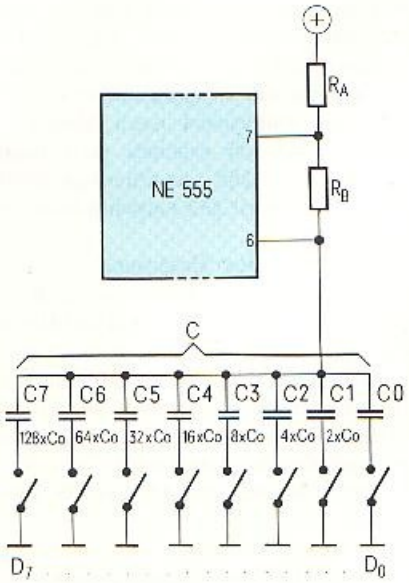
① **Synthesizer:** Programmgesteuert verändern die Schalter D0 bis D7 den Wert eines frequenzbestimmenden Bauteils im Oszillator



② **Tonoszillator:** Frequenzbestimmend sind die Widerstände R_A, R_B und der Kondensator C

Frequenzbereich nach eigenem Ermessen gewählt werden. Nachteilig ist nur, daß Kondensatoren normalerweise nicht so fein abgestuft erhältlich sind wie Widerstände.

Um den Tonoszillator wunschgemäß zu dimensionieren, wurde ein Programm entwickelt (*Bild 4*), das acht Kapazitätswerte aus der E-6-Reihe vorgibt. Nach Eingabe von Werten für die Widerstände R_A und R_B wird die höchste und tiefste Frequenz sowie die zu den einzelnen Datenworten gehörenden Frequenzwerte angezeigt. Da die Kapazitätswerte sich nicht genauso staffeln lassen wie die Gewichtungsfaktoren, kommt es zwangsläufig zu Ungenauigkeiten, die jedoch kaum stören.

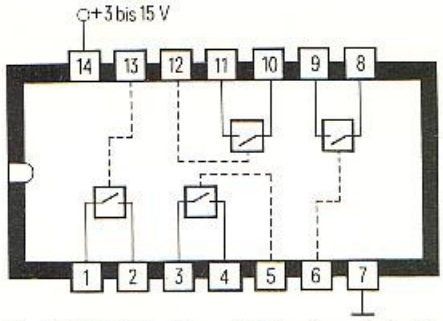


③ **C-Umschaltung:** Die resultierende Kapazität C kann abhängig von den Schalterstellungen 255 verschiedene Werte annehmen

CMOS-Schalter legen die Kondensatoren auf Masse

Das Umschalten der Kondensatoren könnten im Prinzip Relais übernehmen. Allerdings ist dann die Schaltgeschwindigkeit nur gering, und es kann zu Störungen durch Kontaktprellen kommen.

CMOS-Schalter, z. B. der Vierfachschalter 4066 (*Bild 5*), sind die bessere Lösung. Dann muß aber berücksichtigt werden, daß der EIN-Widerstand der einzelnen Schalter rd. 90 Ω beträgt. Damit dieser Wert die berechneten Frequenzwerte nicht wesentlich verschiebt, sollten die Widerstände R_A und R_B (bzw. die entsprechenden Gesamtwiderstände) groß gegenüber dem EIN-Widerstand der Schalter sein. Die Steuereingänge der Schalter sind TTL-kompatibel und können direkt mit dem Ausgangsport der I/O-Schnittstelle verbunden werden.



⑤ **CMOS-Schalter:** H-Pegel an den einzelnen Steuereingängen schließt die Schalter des CD 4066

Im nächsten Teil wird die komplette Schaltung einschließlich Platinenlayout vorgestellt. Die Steuerprogramme, z. B. für eine Miniorgel, sind dann besonders einfach.

Oskar Merker

(*Wird fortgesetzt*)

ZX-Schaltpläne:

Mit 5 DM sind Sie dabei

Bei der J. Schumpich GmbH, 8012 Ottobrunn, gibt es für jeweils 5 DM in Briefmarken Schaltpläne des ZX 81 bzw. ZX Spectrum (Fotokopie von Issue II mit Handeintragungen!). Die Firma bietet außerdem einen Reparaturservice, wobei die Kosten für den Kunden nicht mehr wie früher pauschal ca. 75 DM betragen, sondern jetzt von der Art des Fehlers abhängen. Leider wird der Reparaturservice nur den Kunden geboten, die ihren ZX-Computer bei der Firma Schumpich gekauft haben. -ll

```
   5 FAST
  10 DIM A(8)
  20 DIM B(8)
  30 LET Z=1
  40 LET K=0
  50 LET A(1)=1.0
  60 LET A(2)=2.2
  70 LET A(3)=4.7
  80 LET A(4)=10
  90 LET A(5)=22
 100 LET A(6)=47
 110 LET A(7)=100
 120 LET A(8)=220
 200 PRINT "C-WERTE IN NANO-FARA
D:"
 300 FOR N=1 TO 8
 400 PRINT A(N);" ";
 500 LET A(N)=A(N)*10**-9
 600 NEXT N
 700 PRINT
 800 PRINT
 810 SLOW
 900 PRINT "R(A) = ";
1000 INPUT RA
1100 PRINT RA;" K-OHM"
1200 LET RA=1000*RA
1300 PRINT "R(B) = ";
1400 INPUT RB
1500 PRINT RB;" K-OHM"
1600 LET RB=1000*RB
1605 LET C=0
1610 FOR N=1 TO 8
1615 LET C=C+A(N)
1616 NEXT N
1617 PRINT
1620 LET F=.001*INT (.5+1.44/((R
A+2*RB)*C))
1625 PRINT "F(MIN) = ";F;" KHZ"
1630 LET F=.001*INT (.5+1.44/((R
A+2*RB)*A(1)))
1635 PRINT
1640 PRINT "F(MAX) = ";F;" KHZ"
1695 FOR N=1 TO 150
1699 NEXT N
1700 CLS
1800 FOR G=0 TO 6
1801 FOR M=1 TO 40
1900 LET J=G*40+M
2000 FOR N=8 TO 1 STEP -1
2100 LET K=J/2
2200 LET B(N)=A(9-N)*(J-2*INT K)
2300 LET J=INT K
2400 NEXT N
2500 LET C=0
2600 FOR N=1 TO 8
2700 LET C=C+B(N)
2800 NEXT N
2900 LET F=.001*INT (.5+1.44/((R
A+2*RB)*C))
3000 LET X=M
3100 LET Y=0
3200 IF M<=20 THEN GOTO 3500
3300 LET X=M-20
3400 LET Y=17
3500 LET Z=4
3600 IF F<100 THEN LET Z=5
3700 IF F<10 THEN LET Z=6
3800 IF F<.1 THEN LET Z=7
3900 PRINT AT X,Y;G*40+M;AT X,Y+
Z;F;AT X,Y+12;"KHZ"
4100 NEXT M
4200 FOR F=0 TO 30
4300 NEXT F
4400 CLS
4500 NEXT G
```

④ **Hilfsprogramm:** Nach Eingabe der Werte für R_A und R_B werden die zugeordneten 255 Frequenzwerte angezeigt

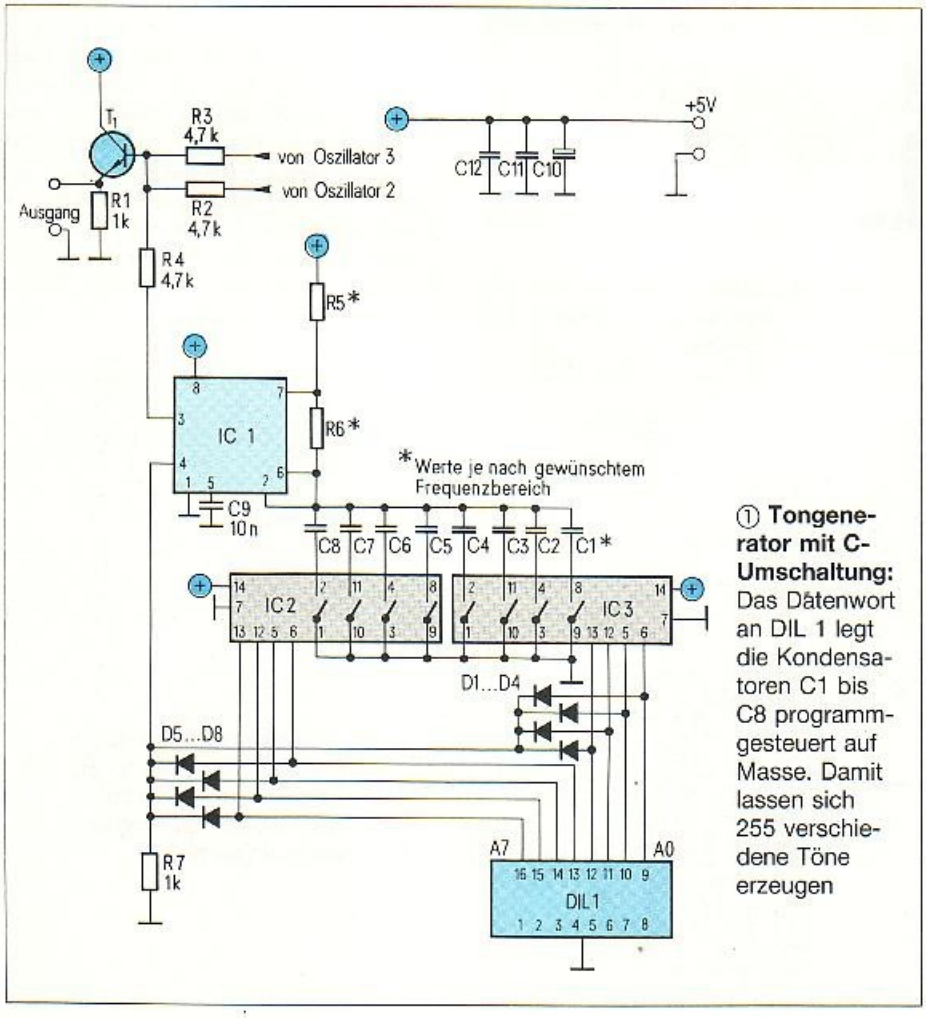
ZX 81 à la carte (9):

Hardware voller Klang

Teil 2: Drei Tongeneratoren auf einen Streich

Wer den hier beschriebenen dreifachen Tongenerator nachbaut, ist auf dem besten Wege, den ZX 81 zum Tonmeister zu machen.

Im vorangegangenen Teil wurde das Prinzip der geschalteten Tongeneratoren erläutert. Hier wird nun die Platine vorgestellt. Sie trägt drei gleiche Tongeneratoren, so daß vielfältige Klangeffekte möglich sind. Zum Betrieb wird die Grundplatine (Heft 13/1983) oder der I/O-Port (Seite 18) benötigt. Da hiermit nur ein Tongenerator betrieben werden kann, erscheinen später Schaltung und Platine eines Sechsfach-Ausgabeports, ausreichend für zwei Exemplare der hier vorgestellten Platine.



① **Tongenerator mit C-Umschaltung:** Das Datenwort an DIL 1 legt die Kondensatoren C1 bis C8 programmgesteuert auf Masse. Damit lassen sich 255 verschiedene Töne erzeugen

Datenwort Null legt die Tongeneratoren still

In *Bild 1* ist die Gesamtschaltung eines Tongenerators gezeigt. Die acht Kondensatoren werden von den CMOS-Schaltern IC 2 bzw. IC 3 in Abhängigkeit von dem an der Eingangsbuchse DIL 1 angelegten Datenwort nach Masse geschaltet. Dadurch ergeben sich insgesamt $2^8 - 1 = 255$ verschiedene Werte für C_{ges} und damit 255 verschiedene Frequenzen.

Liegt dann das Datenwort Null an DIL 1, so sind alle Kondensatoren abgeschaltet. Infolge unvermeidlicher Schaltkapazitäten wirkt aber doch noch ein „Kondensator", so daß die Schaltung auf einer hohen Frequenz schwingen würde. Darum wird bei dem Eingabewert Null der Oszillator über Pin 4 (IC 1) mit dem Diodengatter D1 bis D8 abgeschaltet.

Ist nur eine der Datenleitungen auf H-Pegel, so erhält auch Pin 4 des Oszillators H-Pegel (ODER-Verknüpfung) und der Oszillator schwingt. Erst wenn alle Datenleitungen L-Pegel führen, liegt auch an Pin 4 (Reset-Eingang des IC 1) L-Pegel und der Oszillator ist gesperrt, d. h., es wird kein Ton erzeugt.

Die Frequenzen lassen sich mit dem in Heft 8 vorgestellten Basic-Programm berechnen, allerdings ergeben sich infolge von Bauteiletoleranzen kleine Abweichungen. Ein Frequenzmesser ist also ein durchaus nützliches Meßgerät zum Abgleich und bei der Programmerstellung.

Der Ausgang des Oszillators (Pin 3) führt über einen Entkopplungswiderstand auf die Basis des Ausgangstransistors T1. Seine Basis erhält gleichzeitig auch die Signale der beiden anderen Oszillatoren, so daß am Ausgang das gemischte Signal abgenommen werden kann. Ist eine Mischung nicht erwünscht (z. B. um einzelne Signale getrennt weiterzuverarbeiten), so können die Signale an den im Bestückungsplan mit einem Sternchen gekennzeichneten Stellen

auch einzeln ausgekoppelt werden. T1 sowie R1 bis R4 dürfen dann entfallen.

Aufbau und Inbetriebnahme

Die Schaltung ist auf einer einseitig kaschierten Platine aufgebaut (*Bild 2*). Die Lage und die Werte der Bauteile gehen aus dem Bestückungsplan (*Bild 3*) hervor. Besonders hingewiesen sei auf die vier Kurzschlußbrücken (nicht vergessen!).

Die 5-V-Stromversorgung kann von der Grundplatine aus erfolgen, es muß aber auf eine ausreichende Kühlung des dort montierten Regler-ICs geachtet werden. Auch der I/O-Port (Teil 5 der „à-la-carte-Serie") kann die Stromversorgung selbst übernehmen.

Beim Zusammenschalten von Tongeneratorplatine und Grundplatine ist darauf zu achten, daß die jeweils masseführenden Leitungen richtig miteinander verbunden werden, andernfalls sind die Port-Ausgänge kurzgeschlossen. Vergleicht man die Belegung von DIL 1 mit der entsprechenden Port-Ausgangsbuchse, muß man nämlich berücksichtigen, daß bei einer Steckerdrehung um 180° die Pins 1 bis 8 den Pins 9 bis 16 entsprechen.

Für die Verbindung zur Grundplatine am besten geeignet ist ein 16adriges Flachkabel mit Steckverbindern in Schneid-Klemm-Technik. Das erspart mühsames Löten und vermeidet Verwechslungen bei den Leitungen.

Ein Test des Tongenerators kann mit dem folgenden kleinen Programm durchgeführt werden (hochohmigen Lautsprecher oder Telefonhörerkapsel am Ausgang anschließen):

```
100 FOR N=0 TO 255
200 POKE 23558,N
300 NEXT N
400 POKE 23558,0
```

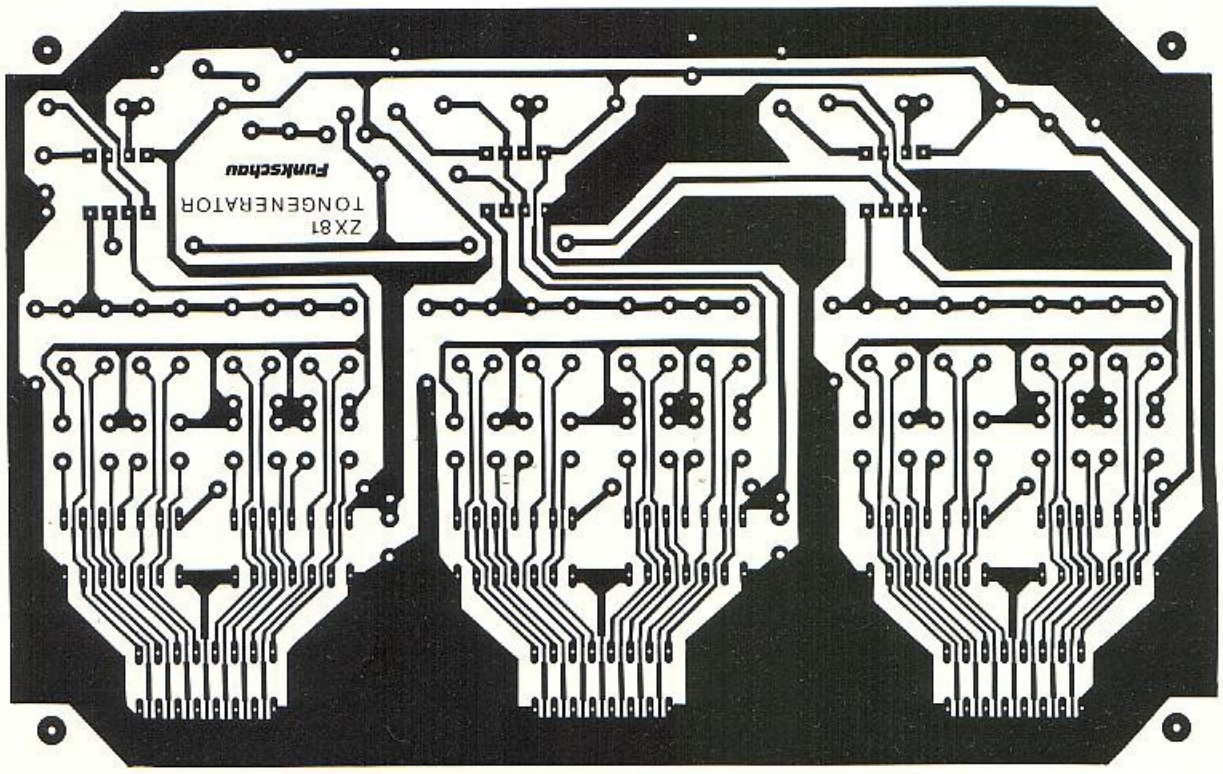
Es muß sich eine von hohen zu tiefen Tönen abgestufte Frequenzfolge ergeben, wobei die Frequenzsprünge zu Beginn am größten sind.

Der Computer zirpt wie eine Grille

Die Tongeneratorplatine ist so recht etwas zum Experimentieren. So läßt sich der ZX 81 beispielsweise als Musikinstrument programmieren. Als Anregung zu eigenen Versuchen ist das folgende Beispiel gedacht:

```
100 IF INKEY$ <> " " THEN GOTO 300
200 POKE 23558,0
300 LET A=CODE INKEY$
400 IF A=XX THEN POKE 23558,YY
:
ZZZ GOTO 100
```

Dabei symbolisiert XX die gewünschte Taste und YY den Datenwert für den gewünschten Ton (ausprobieren, der Tonbereich des Oszillators muß selbstverständlich hardwaremäßig so gelegt sein, daß alle gewünschten Töne erzeugt



② **Platinenlayout:** Wer nur einen Tongenerator benötigt, kann das Layout einfach reduzieren, denn bis auf die Mischstufe sind alle drei Generatoren gleich

werden können). ZZZ ist die Zeilennummer der letzten Programmzeile.

Im Gegensatz zur Software-Tonerzeugung kommt das eigentliche „Orgelprogramm" hier ohne Maschinenprogramm aus. Außerdem ist durch ein geeignetes Programm auch mehrstimmiges Spielen möglich. Hierzu werden aber weitere Ausgabeports benötigt, für die in einem späteren Teil der Serie eine Platine vorgestellt wird.

Klang-Effekte können ebenfalls ausprobiert werden. Das folgende Programm ist auf eine Oszillator-Beschaltung mit den Werten R5 = 470 Ω, R6 = 15 kΩ (Kondensatoren wie angegeben) abgestimmt.

```
100 FOR N = 10 TO 15
200 POKE 23558,N
300 NEXT N
400 GOTO 100
```

Klang Grillenzirpen schon einmal schöner? (Nur beim Betrieb mit Grundplatine: Ablauf im Fast-Modus.)

Oskar Merker

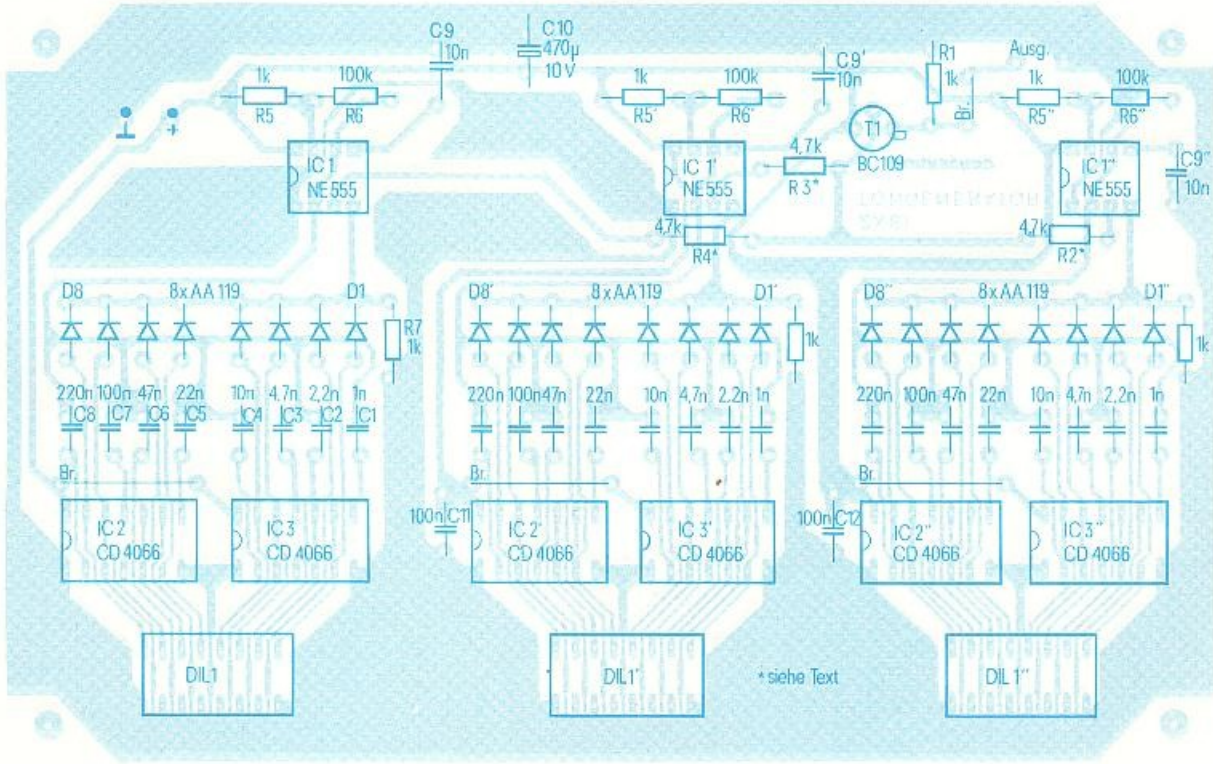
Betrieb mit dem PIO-Port

Soll der PIO-Port die Tongeneratoren steuern, dann ist das PIO-IC in die Betriebsart Datenausgabe zu bringen. Das entsprechende Maschinenprogramm wurde in Teil 7 der „à-la-carte-Serie" veröffentlicht (Heft 5/1984, Seite 70, Bild 2). Die im vorliegenden Beitrag gezeigten Basic-Programme müssen zudem geringfügig geändert werden:

Sorgt mit der Grundplatine der Befehl POKE 23558,N dafür, daß das Datenwort N ausgegeben wird, so fordert der PIO-Port für den gleichen Zweck zwei Befehle: POKE 16514+5,N bringt zunächst im fünften Byte des Maschinenprogramms den Wert N unter, der anschließend mit der Anweisung RAND USR 16514 ausgegeben wird. Das Testprogramm sieht dann so aus:

```
100 FOR N=0 TO 255
200 POKE 16514+5,N
300 RAND USR 16514
400 NEXT N
500 POKE 16514+5,0
600 RAND USR 16514
```

Selbstverständlich ist zuvor noch das Maschinenprogramm zur Datenausgabe gemäß Teil 7 der Serie in einem REM-Kommentar unterzubringen. In einigen Fällen kann es vorkommen, daß der H-Ausgangspegel des PIO-Ports nicht hoch genug ist, um die CMOS-Schalter zu schließen. Dann ist eine nichtinvertierende Pufferstufe zwischen Port-Ausgang und Tongenerator-Eingang zu schalten.



③ **Bestückungsplan:** Die frequenzbestimmenden Bauteile C1 bis C8 sowie R5 bis R6 sind hier beispielhaft dimensioniert. Anstelle der Germaniumdioden können auch Siliziumdioden (1N4148) verwendet werden

ZX 81 à la carte (10):

Telefoncomputer

Außer der Basisplatine zur „à-la-carte"-Serie sind nur wenige Bauelemente nötig, um aus dem ZX 81 einen Telefoncomputer mit Kurzwahl und Wahlwiederholung zu machen.

Damit erst gar keine Mißverständnisse aufkommen: Die hier vorgestellte Schaltung darf nur an nicht amtsberechtigten Nebenstellenanlagen betrieben werden. Wer sie ans öffentliche Fernsprechnetz anschließt, der riskiert Ärger mit der Post!

Vieltelefonierern wird der ZX-81-Telefoncomputer gewiß eine Erleichterung bringen, denn er bietet außer der normalen Handwahl noch die Kurzwahl und Wahlwiederholung. Dieses Leistungsvermögen wird allein von der Software bestimmt. Zuerst wollen wir uns jedoch um die Hardware kümmern.

Keine Qual der Wahl

Angeschlossen wird die kleine Schaltung (*Bild 1*) an den Ausgabe-Port der Basisplatine aus Heft 12/13 1983. Selbstverständlich steht auch einem Anschluß an eine andere Parallel-Schnittstelle nichts im Wege (z. B. PIO-Port), doch muß dann im Steuerprogramm (*Bild 2*) die Port-Adresse (Zeile 10) entsprechend geändert werden. Wird der PIO-Port verwendet, ist außerdem ein Maschinenprogramm zur Datenausgabe hinzuzufügen (siehe auch Seite 23 in diesem Heft).

Von den acht Ausgängen des Ports benötigen wir nur die beiden niederwertigsten, nämlich D0 und D1. Über D0 werden die softwaremäßig erzeugten Wählimpulse ausgegeben, die über T1 das Wählrelais aktivieren. Weil dann das Telefon allerdings im Takt der Wählimpulse mitläuten würde, wird es während des Wählens mit Relais 2, das von D1 gesteuert wird, kurzgeschlossen.

Damit das Telefon auch bei abgeschaltetem Computer noch funktioniert, müssen die Relais Modelle mit Ruhekontakt sein. Deshalb sind Reedrelais nicht geeignet, Kammrelais auch nicht, da sie oft zu träge und überdies laut sind. Am besten verwendet man Miniaturrelais für Platinenmontage.

Software prägt unseren Telefoncomputer

Die Betriebsspannung der Relais sollte zwischen 5 V und 9 V liegen. Ausführungen mit geringer Leistungsaufnahme können unmittelbar vom Rechner gespeist werden (5 V oder 9 V), anderenfalls ist eine externe Spannungsversorgung notwendig.

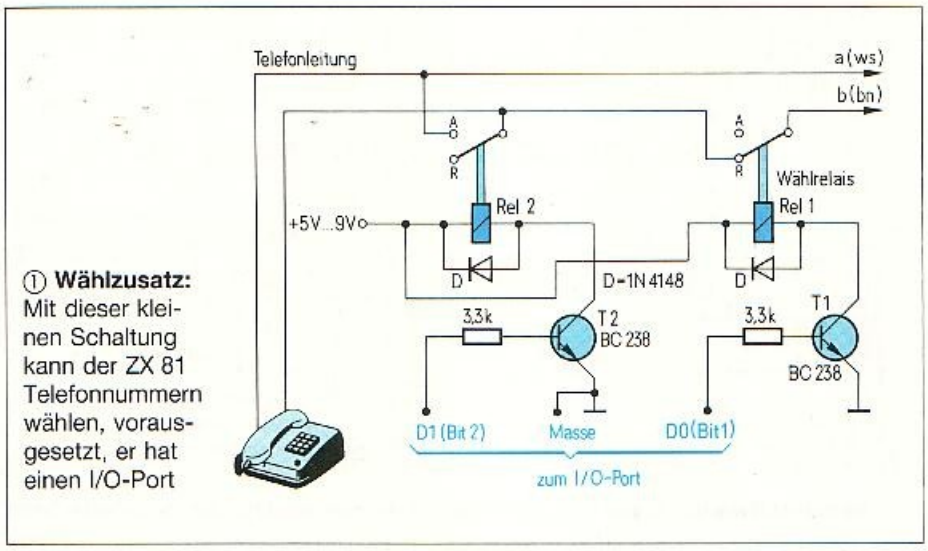
Das Programm, das den ZX 81 zum Telefoncomputer macht, beschert ihm alle die eingangs erwähnten Leistungsmerkmale, die auch kommerzielle Telefoncomputer bieten. Wenn man das Programm eingetippt und mit RUN gestartet hat, erscheint zuerst das von Zeile 100 und 110 vorgegebene „Menü" am Bildschirm.

Entscheidet man sich für die Handwahl, so lassen sich die Ziffern wie bei einem normalen Tastentelefon eintippen. Der Rechner beginnt jedoch erst nach der Bestätigung durch NEWLINE mit dem Wählen, so daß man Tippfehler ausbügeln kann, ohne fremden Leuten auf die Nerven zu gehen.

Die Wahlwiederholung wählt nach Drücken der Taste W die zuletzt gewählte Nummer noch einmal; sie ist also genau das Richtige bei chronisch besetzten Leitungen!

Die Kurzwahl schließlich bietet zugleich ein kleines Telefonregister: Nach Drücken der Taste K werden maximal 40 eingetragene Namen (jeder höchstens 12 Buchstaben) zur Auswahl angezeigt. Gibt man nun eine Nummer von 1 bis 40 ein (durch Eingabe von 0 kann man Einträge ändern oder neue vornehmen), so wird automatisch die zugehörige komplette Rufnummer gewählt.

Den guten Leistungsmerkmalen steht leider ein großer Nachteil gegenüber: Unser Telefoncomputer ist normalerweise auf ein Fernsehgerät angewiesen. Weniger schlimm ist, daß das Aufblättern des Telefonregisters selbst im FAST-Modus ziemlich lange dauert. Hat man jedoch bei der Bedienung des Geräts eine traumwandlerische Sicherheit erlangt, dann darf man durchaus das Fernsehgerät ausschalten, und die Eingaben „blind" vornehmen. In diesem Fall empfiehlt es sich, das Register abzu-



① **Wählzusatz:** Mit dieser kleinen Schaltung kann der ZX 81 Telefonnummern wählen, vorausgesetzt, er hat einen I/O-Port

schreiben bzw. mit COPY ausdrucken zu lassen. Durch Einfügen der Programmzeile 2005 GOTO 2070 ist es überdies möglich, den zeitraubenden Registeraufbau zu unterdrücken.

Mit den Programmzeilen 2087 bis 2100 werden Leerstellen nach denjenigen Ziffern entfernt, die im Kurzwahlregister mit weniger als 15 Ziffern gespeichert sind. Das Wählprogramm würde sonst mit einer Fehlermeldung reagieren. Die Zeilen 3000 bis 3130 ermöglichen Änderungen bzw. Neueinträge im Register.

Das eigentliche Wählprogramm beginnt mit Zeile 4000. Ab hier (bis Zeile 4300) werden die Wählimpulse mit PAUSE-Befehlen erzeugt und über die POKE-Befehle an den Port ausgegeben. POKE P,3 läßt beide Relais anziehen; Relais 2 bleibt dann während des gesamten Wählvorgangs aktiviert. Nur Relais 1 reagiert auf POKE P,2 indem es abfällt. Der PAUSE-Befehl in Zeile 4120 bestimmt die Dauer der einzelnen Wählimpulse; die Pause zwischen den Wählimpulsen wird mit Zeile 4150 festgelegt. Die von Zeile 4250 bestimmte Wartezeit ist die Pause zwischen den einzelnen Ziffern einer gewählten Nummer.

Die Dauer der Pausen in Zeile 4120 und 4150 wurde experimentell ermittelt. Die Werte sind nicht ganz normgerecht. Als Orientierung für eigene Experimente gilt: Die Öffnungsdauer des Wählrelais sollte je Impuls zwischen 51 ms und 72 ms liegen, die Schließungsdauer zwischen 31 ms und 48 ms. Das zulässige Tastverhältnis reicht damit von 1,3 : 1 bis 1,9 : 1. Eventuell sind die Werte der beiden PAUSE-Befehle geringfügig zu ändern, damit Verbindungen fehlerfrei zustandekommen. Das Flackern des Bildes während der Ausführung der PAUSE-Befehle ist zugleich eine optische Kontrolle für den Programmablauf.

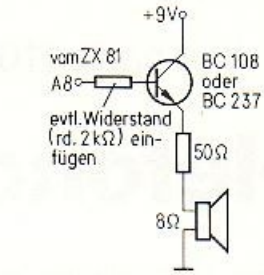
Ist ein Register aufgebaut worden, so läßt es sich durch Betätigen der Taste S auf Kassette speichern; der dazugehörige Programmteil beginnt mit Zeile 9900. Der letzte Befehl dort lautet GOTO 100. Er bewirkt einen automatischen Start des Programms nach dem Laden von Kassette. Wird das Programm versehentlich – oder um Änderungen vorzunehmen – unterbrochen, so darf es nur mit GOTO 100 erneut gestartet werden. Jeder Start mit RUN oder GOTO 10 löscht sämtliche Einträge (Variablen) im Register.

Wolf-Dieter Roth/-ll

ZX-81-Hardwaretip:

Tastatur-Pieps

Die im *Bild* angegebene einfache Schaltung erlaubt eine akustische Kontrolle des ZX 81. Das an der Adreßleitung A8 anliegende Signal wird einfach abgegriffen, verstärkt und einem Kleinlautsprecher zugeführt. Die Adreßleitung kann auch hinter der Diode (im ZX-Schaltplan D6) an der Tastatur angezapft werden, was sich als praktisch beim Einbau in eine Zusatztastatur erweist. Die Schaltung kann vom ZX-Netzteil mitversorgt werden; Masse und + 9 V lassen sich am Spannungsregler abgreifen.



Tastatur-Pieps: Hörbar gemacht, meldet das Signal A8 des ZX 81 eine Tastenbetätigung

Jeder Tastendruck ergibt ein akustisches Echo, sobald der Computer die Eingabe erkannt hat; daher kann man nun auf der Folientastatur etwas eintasten, ohne dabei immer wieder auf den Bildschirm schauen zu müssen. Basic- und Maschinensprache-Programme verursachen typische Geräusche, so daß man auch hören kann, ob ein Programm abgestürzt ist. Der Haken an dieser „Minimallösung" ist, daß durch das Signal auf A8 der Lautsprecher ein ständiges Geräusch erzeugt, aus dem ein Tastendruck lediglich als kurzes „Plop" herauszuhören ist – aber immerhin...

Michael Schramm

```
  10 LET P=23558
  20 DIM V$(40,12)
  30 DIM D$(40,15)
 100 PRINT "H = HANDWAHL",,,,"K
= KURZWAHL",,,,"U = WIEDERHOLUNG
",,,"B = BEENDEN (OHNE SPEICHERN
),,,"S = SPEICHERN"
 110 PRINT AT 15,0;"BITTE ENTSPR
ECHENDE TASTE      DRUECKEN"
 120 IF INKEY$="H" THEN GOTO 100
0
 130 IF INKEY$="B" THEN STOP
 140 IF INKEY$="S" THEN GOTO 990
0
 150 IF INKEY$="K" THEN GOTO 200
0
 160 IF INKEY$="U" THEN GOTO 150
0
 200 GOTO 120
1000 CLS
1020 PRINT "GEBEN SIE DIE NUMMER
 OHNE          ZWISCHENRAUM EIN"
1030 INPUT N$
1040 GOSUB 4000
1050 CLS
1060 GOTO 100
1500 CLS
1503 PRINT AT 5,3;"ICH WAEHLE
 N$
1505 GOSUB 4000
1510 GOTO 100
2000 CLS
2010 FAST
2020 PRINT " 0 AENDERN"
2030 FOR F=1 TO 40
2040 IF F<10 THEN PRINT " ";F;"
 "
2050 IF F>=10 THEN PRINT F;" ";
2060 PRINT V$(F);
2065 NEXT F
2068 SLOW
2070 INPUT F$
2080 IF F$="0" THEN GOTO 3000
2085 CLS
2087 LET N$=" "
2090 LET L$=D$(VAL F$)
2092 LET S=LEN L$
2094 IF L$(S)=" " THEN LET L$=L$
(1 TO S-1)
2096 LET S=S-1
2098 IF L$(S)=" " THEN GOTO 2094
2099 LET N$=L$
2100 PRINT "ICH WAEHLE NR. ";N$;
,,"TEILNEHMER ";F$,,,,V$(VAL F$)
2110 GOSUB 4000
2120 GOTO 100
3000 PRINT AT 0,0;"WELCHEN TEILN
EHMER (NR. 1-40)?"
3010 INPUT H$
3020 CLS
3030 PRINT "TEILNEHMER ";H$,,,,"
NR. ";D$(VAL H$),,,V$(VAL H$)
3040 PRINT
3050 PRINT "NEUE NUMMER?"
3060 INPUT J$
3070 LET D$(VAL H$)=J$
3080 PRINT
3090 PRINT "NEUER NAME?"
3100 INPUT K$
3110 LET V$(VAL H$)=K$
3120 CLS
3130 GOTO 100
4000 POKE P,2
4010 PAUSE 25
4020 FOR L=1 TO LEN N$
4030 LET Z$=N$(L)
4040 IF Z$="0" THEN LET Y$="10"
4050 IF NOT Z$="0" THEN LET Y$=Z
$
4100 FOR Y=1 TO VAL (Y$)
4110 POKE P,3
4120 PAUSE 1,9
4140 POKE P,2
4150 PAUSE 1
4200 NEXT Y
4250 PAUSE 25
4300 NEXT L
4310 POKE P,0
4330 PAUSE 25
4400 CLS
4410 RETURN
9900 CLS
9905 PRINT "SCHALTEN SIE DEN REC
ORDER AUF   AUFNAHME"
9910 PAUSE 200
9920 CLS
9930 PRINT "O.K.? DANN DRUECKE "
"NEWLINE"""

9935 IF INKEY$="" THEN GOTO 9935
9940 CLS
9950 SAVE "TELEFONWAEHLAUTOMA"
9960 GOTO 100
```

② **Steuersoftware:** Dieses Programm ist auf die Basisplatine der „à-la-carte-Serie" zugeschnitten

Decker & Computer

Spitzenprodukte für Sinclair-Computer

ZX-ASZMIC-ROM **EPROM-Programmierer**

EPROM-Karte **Autostart**

Dazu gehören natürlich sehr ausführliche Handbücher und Anleitungen.

Neu aus unserer Buchproduktion :

Die RAM-Bedienung beim ZX 81

Eine der interessantesten Entwicklungen der letzten Jahre :

Haben Sie sich schon mal über das angesteckte RAM-Pack geärgert ?

Jetzt gibt es Abhilfe ! Rüsten Sie Ihren Rechner intern auf 16k CMOS- RAM auf !

Haben Sie schon einmal versucht, Maschinenprogramme oberhalb von 32k laufen zu lassen ?

Hier steht, wie Ihr RAM dazu funktionieren muss und wie man die ULA austrickst.

Sind Ihre BASIC-Programme sehr lang und Sie brauchen noch Platz für weitere Programmzeilen ?

Hier steht, wie man das Display- File über die 32k-Kante schieben kann und wie das RAM dazu beschaffen sein muss.

Haben Sie ein 64k-RAM von Downsway oder Kade ?

Hier ist der Schaltplan !

Wissen Sie, was ein Grenzsystem ist ?

Maximal 56k RAM für Maschinen- Programme !

Das kleine ZX-Hardwarebuch

Brandneu, frisch aus der Druckerei. 370 Seiten voller Überraschungen

- Analyse des Centronics-Drucker-Interfaces von Memotech mit Schaltplan und kommentiertem Sourcelisting
- Analyse der Tastatur von Memotech mit Schaltplan
- Analyse der RAMs von Memotech : 16k RAM, 32k RAM und 64k RAM Mit Schaltplänen und Listings der Steuer-PROMs
- Beschreibung von RAM-Platinen
- Centronics-Interface
- eine der ausführlichsten PIO-Beschreibungen fuer ZX 81 und Spectrum
- Tabellen aller I/O-Adressen, Systemsteuerung des ZX 81
- Autostart (siehe auch Bericht im Sinclair-Sonderheft von Happy Computer)
- EPROM-Programmierer (siehe auch Testbericht im Funkschau-Heft Nr. 11/85)
- EPROM-Karten, bis maximal 64k EPROM
- Die ULA, ihre Funktion und ihr Innenleben
- und mehr

mit 7 Fotos, über 50 Tabellen und mehr als 100 Zeichnungen

Preise: RAM-Bedienung DM 30.-- / Hardwarebuch DM 40.--

Den ausführlichen Katalog erhalten Sie für DM 5.-- in Briefmarken

Decker & Computer, Postfach 967, 7000 Stuttgart -1

ZX 81 à la carte (11):

Nachwuchs für den Ausgabe-Port

Die in FUNKSCHAU 12/1983 vorgestellte Grundplatine zu unserer Serie hat nur einen Ausgabe-Port. Der bekommt jetzt gleich sechs ebenbürtige „Kollegen", so daß bis zu sieben Peripheriegeräte zugleich mit Daten versorgt werden können.

Die Grundplatine der ZX-à-la-carte-Serie hat insgesamt drei Schnittstellen zur Außenwelt (*Bild* 1): Einen Ein- und Ausgabe-Port sowie die 24polige DIL-Buchse. Diese DIL-Buchse führt u. a. den Datenbus sowie 13 decodierte Adressen. Dadurch können weitere Ergänzungsschaltungen an die Grundplatine angeschlossen werden. Die Adresse, unter der die jeweilige Schaltung angesprochen wird, darf dabei der Anwender selbst auswählen. Die hier vorgestellte Erweiterung des Ausgabe-Ports nutzt sechs dieser Adressen.

Zwischenspeicher nageln die Daten fest

Bild 2 zeigt das Schaltbild des Sechsfach-Ausgabe-Ports. Es findet dasselbe Schaltungsprinzip Anwendung wie beim Ausgabe-Port auf der Grundplatine. Ein 8-Bit-Zwischenspeicher (Latch) 74LS373 übernimmt hier wie dort die auf dem Datenbus liegende Information, sobald sein Freigabe-Eingang (Enable Pin 11) H-Potential führt.

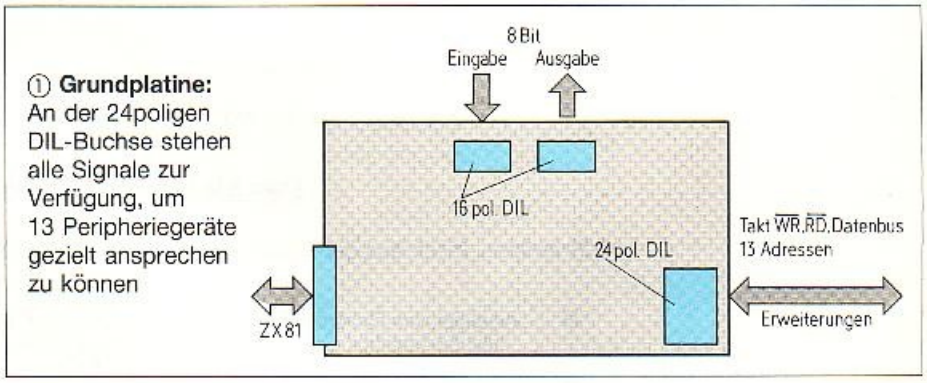
Das soll der Fall sein, wenn die gewünschte Adresse angesprochen und gleichzeitig ein Schreibimpuls $\overline{WR}$ vom ZX 81 geliefert wird. Die hierzu nötige UND-Verknüpfung beider Signale erfolgt mit Hilfe von IC 1, einem sechsfach-Bustreiber 74LS366 (*Bild* 3). Solange seine beiden Enable-Eingänge (Pin 1 und Pin 15) gleichzeitig L-Pegel führen, sind die sechs invertierenden Bustreiber aktiviert.

In der Schaltung ist Pin 15 mit der Schreibleitung verbunden, während Pin 1 ständig an Masse liegt. Die decodierten Adreßsignale werden den Eingängen des 74LS366 zugeführt. Falls nun die gewünschte Adreßleitung angesprochen wird (L-aktiv), steht am jeweiligen Ausgang des Bustreibers ein H-Signal zum Ansteuern des dazugehörigen Zwischenspeicher-ICs zur Verfügung.

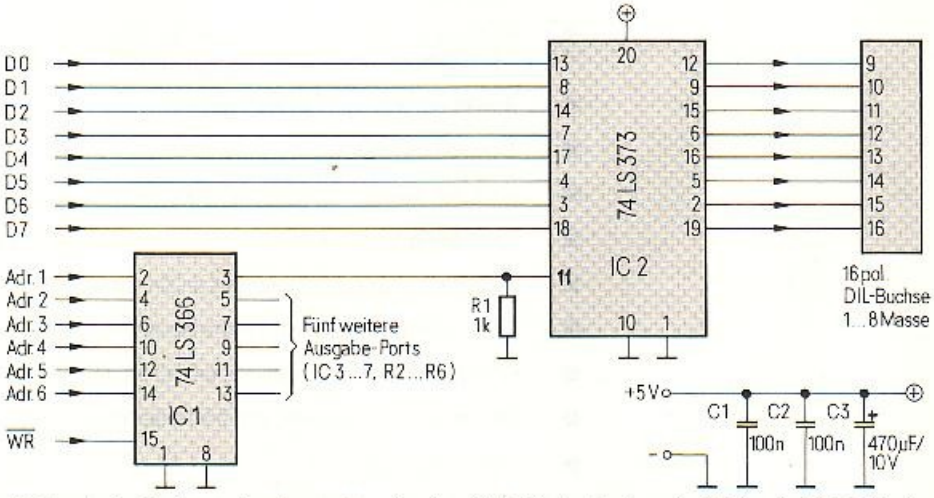
Solange Pin 15 von IC 1 H-Pegel führt, befinden sich die Ausgänge im hochohmigen Zustand. Um zu vermeiden, daß der aus dem TTL-Eingang der Speicher-ICs (Pin 11) herausfließende Strom einen Spannungsabfall und damit ein H-Signal verursacht, sind alle Enable-Eingänge der Zwischenspeicher-ICs über 1 kΩ mit Masse verbunden.

Die Daten-Ausgänge der ICs sind jeweils auf 16polige DIL-Fassungen geführt, an die z. B. die Tongenerator-Schaltungen von Seite 30 über Flachkabel angeschlossen werden können. Dabei ist darauf zu achten, daß die masseführenden Leitungen beider Platinen miteinander verbunden werden.

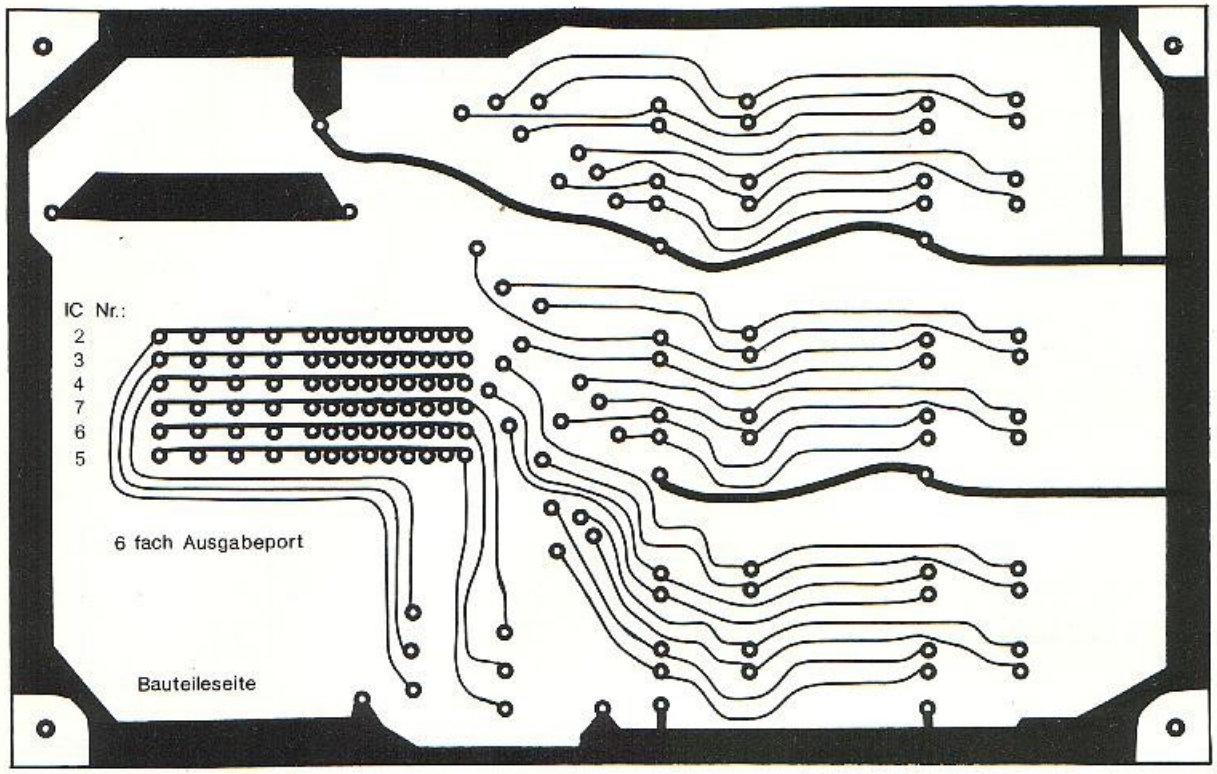
Die Schaltung wird auf einer zweiseitig kupferkaschierten Platine aufgebaut. *Bild* 4 zeigt das Platinenlayout der Löt-



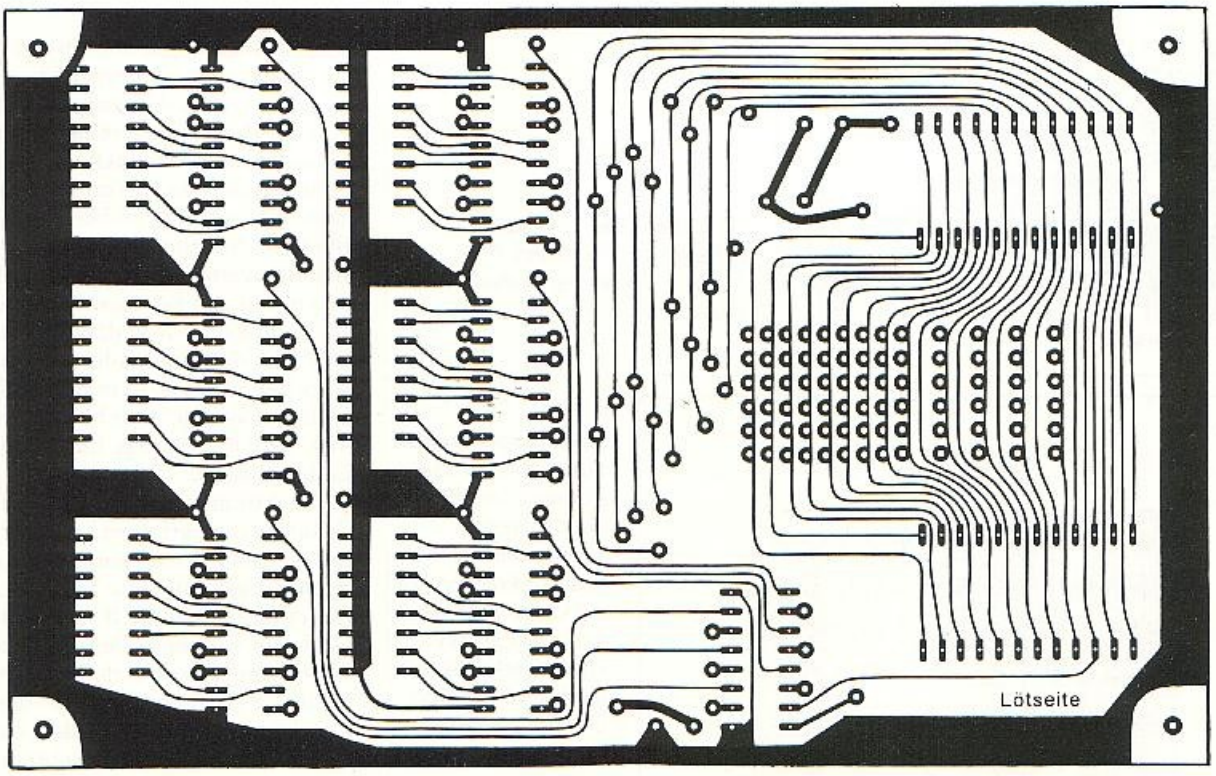
① **Grundplatine:** An der 24poligen DIL-Buchse stehen alle Signale zur Verfügung, um 13 Peripheriegeräte gezielt ansprechen zu können

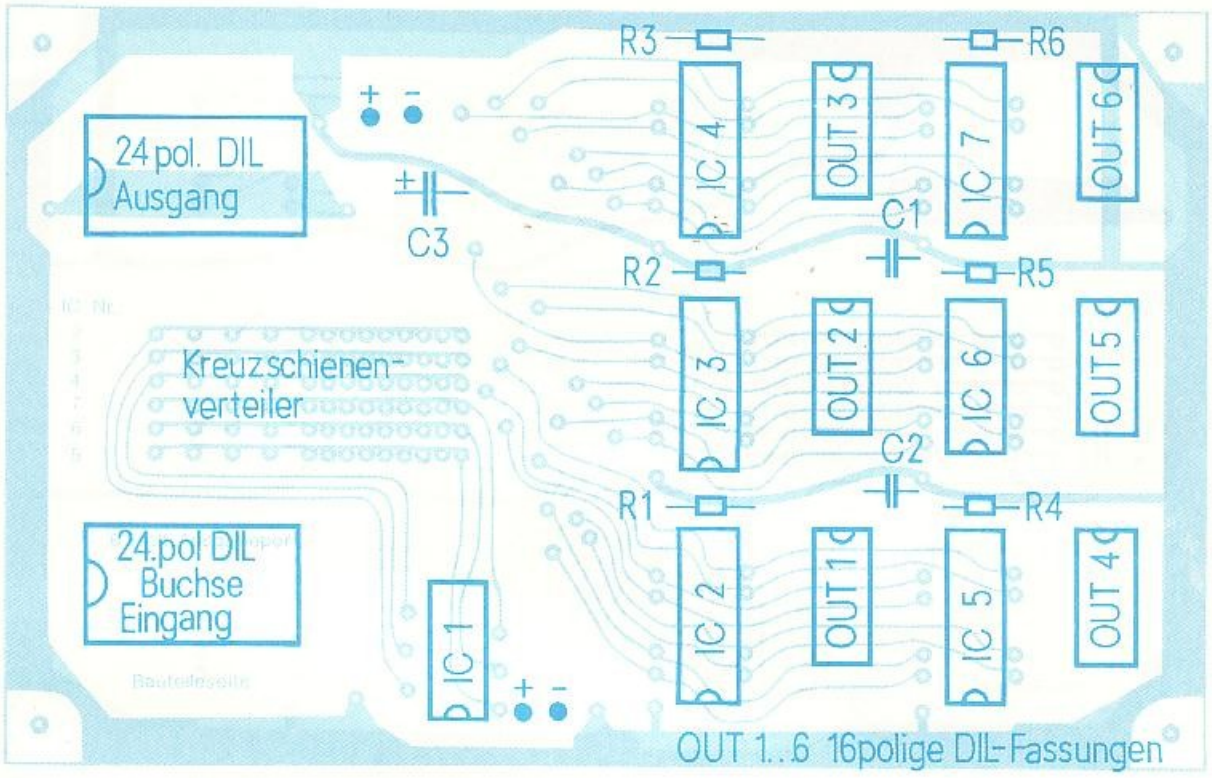


② **Sechsfach-Ausgabe-Port:** Durch eine UND-Verknüpfung in IC 1 wird IC 2 (Zwischenspeicher) nur dann freigegeben, wenn ein decodiertes Adreßsignal und ein Schreibimpuls vorliegen

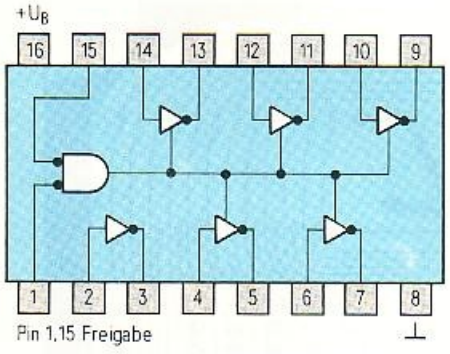


④ **Platinenlayout:** Unten die Lötseite, oben die Leiterbahnen beim Blick auf die Bestückungsseite

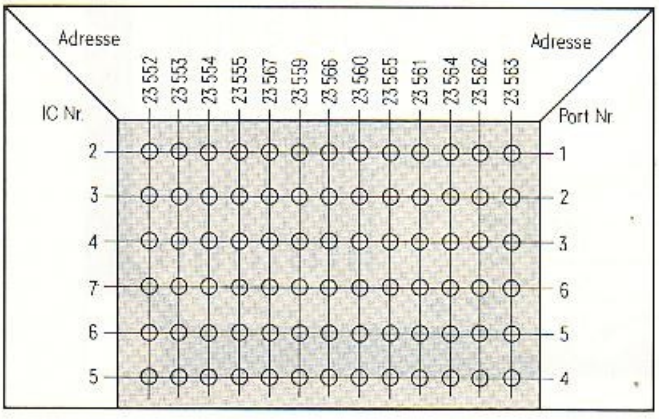




⑤ **Bestückungsplan:** Sechs DIL-Buchsen (DIL: Dual-In-Line) sind die Ausgänge der Ausgabe-Ports



③ **Bustreiber 74LS366:** L-Pegel an Pin 1 und Pin 15 gibt die sechs Pufferstufen frei



⑥ **Adressenzuordnung:** Hier ist das Adressierungsfeld auf der Bestükkungsseite der Platine symbolisiert (13 × 6 Lötaugen). Wird ein Lötauge durchkontaktiert, läßt sich der betroffene Port unter der zugeordneten Adresse aktivieren

und Bestückungsseite, *Bild 5* den Bestückungsplan. Vor dem Einsetzen der Bauteile und IC-Fassungen müssen alle von der Bestückungsseite her sichtbaren Lötaugen durchkontaktiert werden.

Ausgenommen davon sind die Lötaugen zwischen den beiden 24poligen DIL-Buchsen. Sie dienen als eine Art „Kreuzschienenverteiler" zur Auswahl der für den jeweiligen Port gewünschten Adresse. Ein Beispiel mag dies verdeutlichen: Soll IC 2 unter der Adresse 23559 angesprochen werden, so muß in Reihe 1 das sechste Lötauge durchkontaktiert werden. *Bild 6* zeigt das hierzu gehörende Codierschema.

Die Stromversorgung kann über die Grundplatine erfolgen, allerdings muß der dort montierte Spannungsregler ausreichend gekühlt sein. Falls noch weitere Zusatzplatinen (z. B. Tongeneratoren) betrieben werden sollen, ist es aber besser, ein separates Netzteil zu verwenden.

Eine Funktionskontrolle der Schaltung ist rasch möglich, indem der bereits bei der Grundplatine beschriebene, mit Leuchtdioden bestückte Teststecker verwendet wird.

Oskar Merker

ZX-81-Softwaretest:

Tempo, Tempo

Das Schnell-Ladeprogramm GM Save

Mit dem Hilfsprogramm GM Save wird der ZX 81 zum flotten Sprinter, wenn er Anwenderprogramme auf Band speichert und von dort lädt. Zehnfacher Tempogewinn ist doch ein Wort!

Aus Belgien erhielten wir ein Schnell-Ladeprogramm zum Test, das einige Leckerbissen bietet und mit 20 DM zudem preisgünstig ist (Liefernachweis ist bei der Redaktion abzurufen).

Um es gleich vorwegzunehmen: Das Programm GM Save ist sein Geld wert. Es verkürzt den Zeitaufwand zum Speichern eines Programmes auf Band erheblich. Im Test gelang es z. B. ein Programm, das normalerweise eine Speicherzeit von 225 s erforderte, in nur 30 s fehlerfrei aufs Band zu bringen. Selbstverständlich verkürzt sich auch die Ladezeit in gleichem Maße.

GM Save ist ein Maschinenprogramm mit einem kurzen Basic-Teil. Das Maschinenprogramm kann durch Verändern von RAMTOP in jeden Teil des RAM-Speichers geladen werden (Obergrenze: 32768). Damit ist GM Save auch dann lauffähig, wenn man den ZX 81 mit beliebigen Speichererweiterungen bis hin zu 64 KByte betreibt.

Geladen ist das Programm selbst (LOAD "") in rd. 35 s, worauf es sich selbst zur Stelle meldet.

Über dem Wort „Error" muß dann der Wert Null stehen, als Zeichen dafür, daß das Programm fehlerfrei geladen wurde. Anderenfalls ist der Ladevorgang mit anderer Lautstärkeeinstellung zu wiederholen (kann je nach Recorder kritisch sein). Im Test mußte bei einem Recorder sogar die Kopfstellung verändert werden, bevor sich GM Save einwandfrei laden ließ!

Das Programm belegt stets 1 KByte unterhalb des angetroffenen Wertes von RAMTOP und setzt dann seinerseits RAMTOP um 1 KByte tiefer. Außerdem belegt es rd. 0,5 KByte im Programmspeicher. Drückt man jetzt nach dem Laden die Taste BREAK, taucht das Bild am Bildschirm auf.

Durch die in der Bildmitte aufgelisteten vier Variablen läßt sich GM Save in weiten Grenzen eigenen Erfordernissen anpassen. Das beginnt mit der Variablen NBR. Ihr ist hier beispielsweise der Wert vier zugewiesen. Wollte man nun mit GM Save ein Programm speichern, dann geschieht das automatisch viermal hintereinander: Ein Programmverlust wegen einer fehlerhaften Bandstelle wird damit ausgeschlossen. Freilich verlängert jede Kopie die Speicherzeit, so daß man normalerweise den Wert NBR = 2 wählen sollte.

Besonders interessant ist die Variable W. Ihr Wert bestimmt, mit welcher Baudrate gespeichert wird. Beim Wert 23 sind das beachtliche 3603 Baud. Die Original-SAVE-Funktion schafft nur rd. 300 Baud! Weist man mit einer einfachen LET-Anweisung W höhere Werte zu, dann sinkt die Baudrate, bei niedrigeren Werten steigt sie, wobei die Obergrenze praktisch nur durch die Qualität der Magnetbandaufzeichnung gesetzt ist. Hier lohnt sich also der Einsatz von Spulen-Tonbandgeräten.

```
START WITH GOTO 0
      OR    PRINT USR 32384

VARIABLE SETTINGS IN USE :

NUMBER COPYS        NBR     4
WRITE TIMING        W       23
TYPE 1ST COPY       F$      NEG
OTHER COPYS         O$      <>

ACTUAL BAUDRATE IS  3.603

OTHER TIMINGS SET BY  O$ D$ R$
INCR R$ IF MORE ERR ON LOW VOL
```

Variablenmenü: Die Eigenschaften von GM Save lassen sich an das verwendete Tonbandgerät anpassen

Bemerkenswert ist auch die Variable F$. Mit ihr lassen sich jetzt Ladeprobleme lösen, wenn es durch die Elektronik des Tonbandgeräts zu einer Phasendrehung des Signals von 180° kommt. Der Alternativwert zu NEG lautet POS. Die Variable O$ bestimmt schließlich, ob beim Speichern alle Kopien (NBR) mit gleicher oder mit abwechselnd um 180° gedrehter Phasenlage aufgezeichnet werden (wichtig, wenn man Programme von verschiedenen Tonbandgeräten laden möchte).

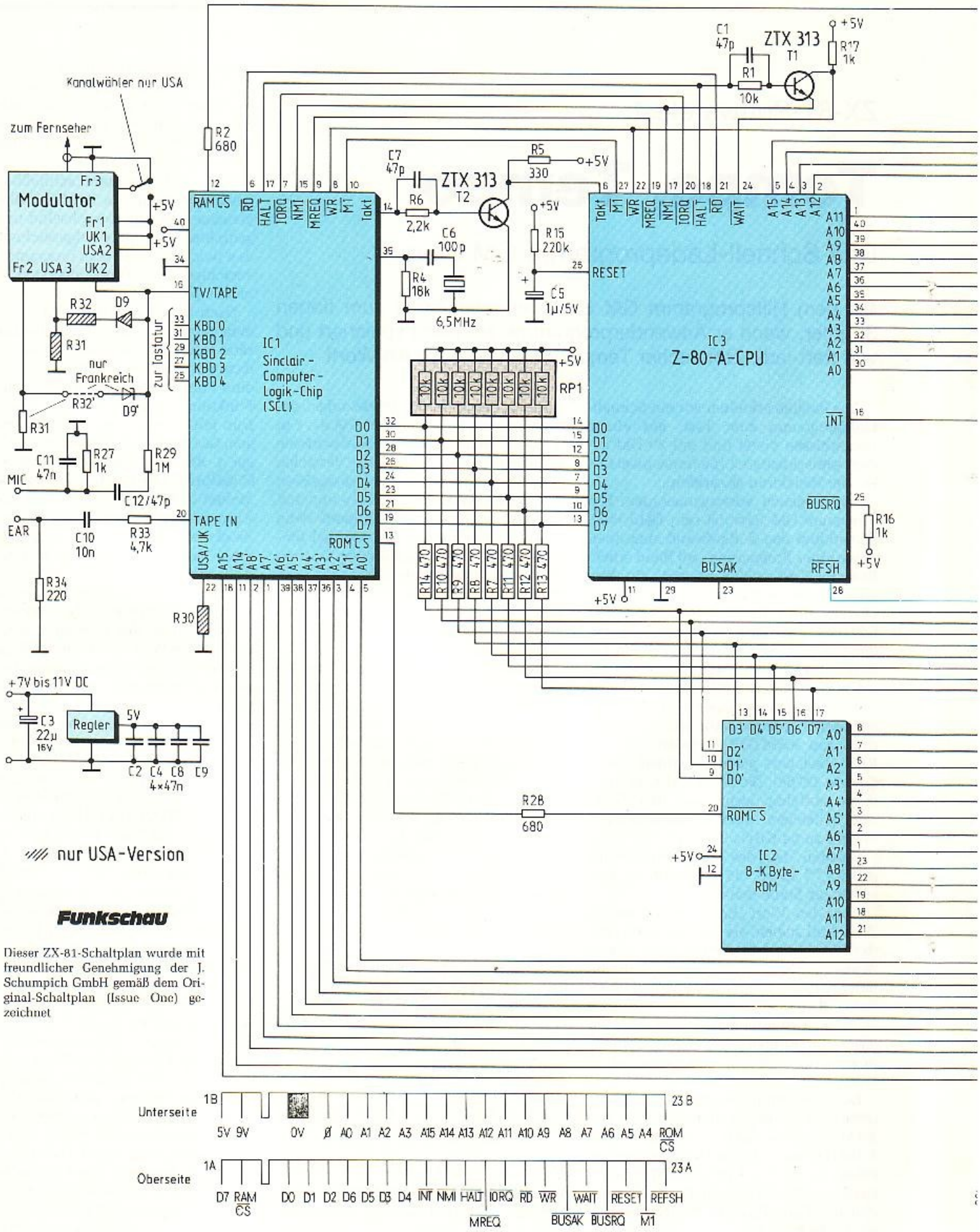
Nach dem Laden von GM Save muß man zuerst die Qualität des Tonbandgerätes feststellen. Mit GOTO 0 starten wir dazu das Programm: Jetzt zeigt GM Save eine Funktionsliste am Bildschirm. Der Funktion SAVE ist Taste 1 zugeordnet. Nun gilt es, eine neue Kassette einzulegen, GM Save darauf abzuspeichern (Taste 1 drücken) und anschließend mit COMPARE (Taste 2) wieder zu laden. Meldet sich das Programm mit Error 0, dann verkraftet das Tonbandgerät 3603 Baud und man kann mit kleinerem Wert für W versuchen, eine noch höhere Baudrate zu erzielen.

Kommt es zu Lesefehlern, muß der Wert von W (wieder) vergrößert werden. Schließlich ist das Programm dann mit dem besten Wert von W endgültig abzuspeichern. Zum Abspeichern von Anwenderprogrammen ist diese Version von GM Save als erstes zu laden. Anschließend kann man sein eigenes Programm eintippen oder mit LOAD von Band laden.

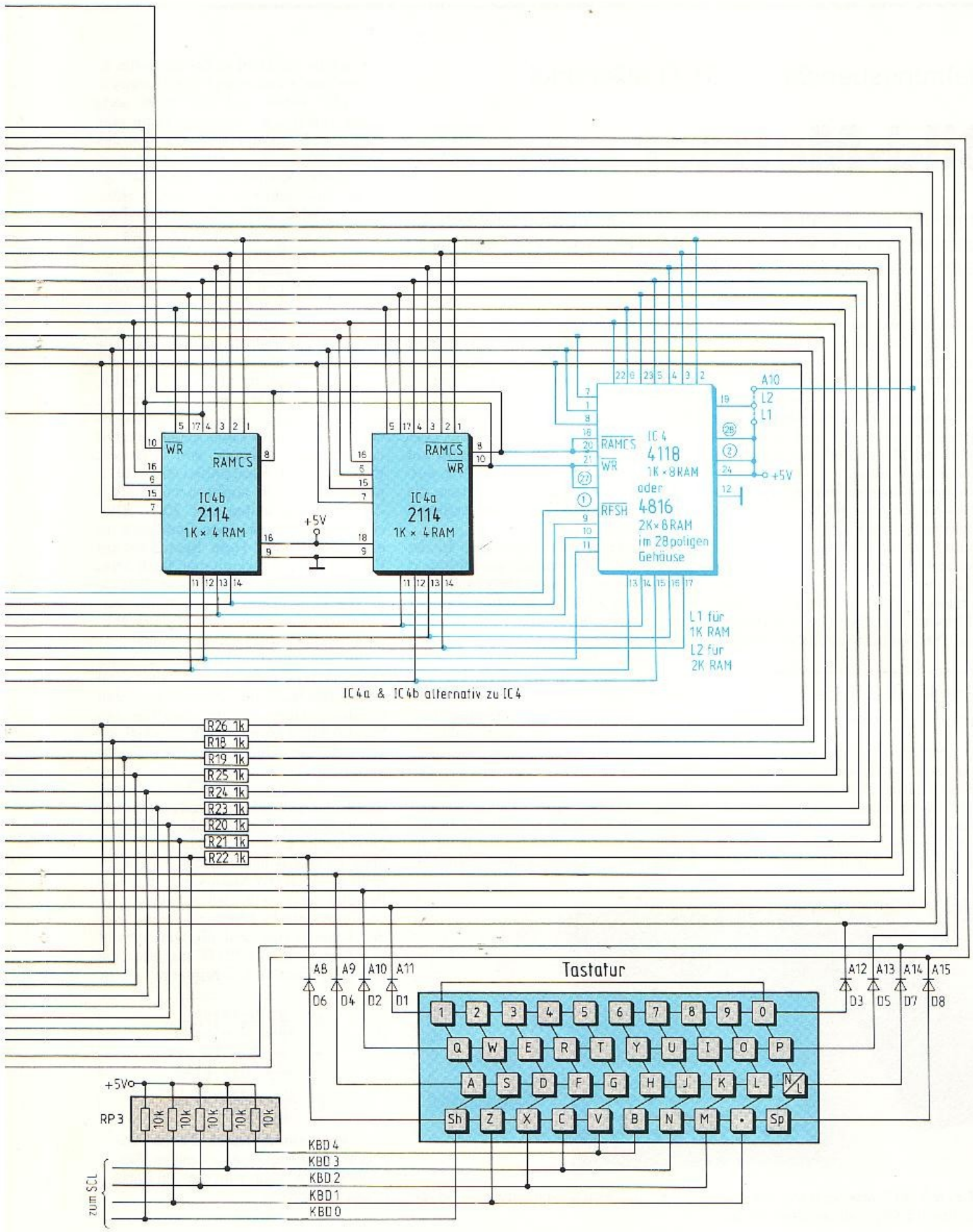
Ist zum Abspeichern in der komprimierten Form alles vorbereitet, wird GM Save mit PRINT USR 32384 aufgerufen. Ein Druck auf Taste 1 speichert daraufhin das Anwenderprogramm mit der gewählten Baudrate. Mit COMPARE läßt sich nun prüfen, ob korrekt gespeichert wurde. Zum Wiedereinlesen genügt die Eingabe von LOAD "" (kein Programmname zulässig!); ein vorheriges Laden von GM Save ist nicht notwendig.

Aus der (deutschen) Bedienungsanleitung sind wir zuweilen nicht recht schlau geworden. So ist z. B. die COMPARE-Funktion mißverständlich beschrieben. Kennern des ZX 81 wird zudem noch eine Funktion geboten, die das komprimierte gemeinsame Speichern verschiedener Maschinenprogramme zulassen soll. Sie ist jedoch unzureichend erklärt und ließ sich nicht mit jedem Recorder nutzen! Das kratzte freilich nur gering am guten Eindruck, den das Programm auf uns machte.

Jürgen Zimmermann/-ll



Dieser ZX-81-Schaltplan wurde mit freundlicher Genehmigung der J. Schumpich GmbH gemäß dem Original-Schaltplan (Issue One) gezeichnet


IC4b
2114
1K × 4 RAM
WR
RAMCS
IC4a
2114
1K × 4 RAM
RAMCS
WR
+5V
IC 4
4118
1K × 8 RAM
oder
4816
2K × 8 RAM
im 28poligen
Gehäuse
RAMCS
WR
RFSH
A10
L2
L1
+5V
L1 für
1K RAM
L2 für
2K RAM
IC4a & IC4b alternativ zu IC4
R26 1k
R18 1k
R19 1k
R25 1k
R24 1k
R23 1k
R20 1k
R21 1k
R22 1k
A8 A9 A10 A11
D6 D4 D2 D1
Tastatur
A12 A13 A14 A15
D3 D5 D7 D8
1 2 3 4 5 6 7 8 9 0
Q W E R T Y U I O P
A S D F G H J K L N/L
Sh Z X C V B N M . Sp
+5V
RP 3
10k 10k 10k 10k 10k
KBD 4
KBD 3
KBD 2
KBD 1
KBD 0
zum SCL

Erfahrungsbericht: ZX-81-Grafikmodul

Bild-Meister

Normalerweise lassen sich mit dem ZX 81 nur grobe Grafiken am Bildschirm zusammenzimmern. Detailreichtum wird erst nach Anschließen eines Grafikmoduls geboten.

Getestet haben wir das HRG-Modul (HRG: High Resolution Grafik) von K-Tronics, das im Vertrieb der Computer Accessoires GmbH, Ottobrunn, bei Redaktionsschluß 179 DM kostete. Zum Lieferumfang gehören das Modul, eine Demonstrationskassette und eine ausführliche leichtverständliche (deutsche) Bedienungsanleitung.

Hardware-Voraussetzung zum Betrieb des Moduls ist eine RAM-Erweiterung von mindestens 16 KByte. Das HRG-Modul muß unmittelbar an der ZX-81-Schnittstelle angeschlossen werden (*Bild 1*); es nimmt dann huckepack (durchgeschleifte Steckerleiste) das RAM-Modul auf. Nach dem Anschluß ist das HRG-Modul sofort betriebsbereit.

Grafikseiten fordern Platz

Mit dem HRG-Modul lassen sich abhängig vom verfügbaren Speicherplatz mehrere „Grafikseiten" gestalten, wobei die Auflösung jeweils 192 × 248 Bildpunkte beträgt. Jede Grafikseite benötigt rd. 6,3 KByte Speicherplatz, der, das ist der erste Schritt, durch Herabsetzen von RAMTOP vorab reserviert werden muß.

Die übliche Bilddarstellung des ZX 81 bleibt von den Grafikseiten völlig unbeeinflußt. Zum normalen Bildspeicher kommt vielmehr ein weiterer hinzu, der jedesmal aufgerufen werden muß, wenn die dort abgelegte Grafik abgebildet oder verändert werden soll. Der Zugriff auf eine Grafikseite ist damit ziemlich aufwendig im Vergleich zu Computern, die bereits serienmäßig eine hohe Grafikauflösung haben.

Ein Beispiel mag das zeigen: In der Mitte des Bildschirms soll ein Bildpunkt gesetzt werden. Ohne HRG-Modul ist das einfach: PLOT 31,22 bildet aber keinen einzelnen Bildpunkt, sondern einen ganzen Klecks ab.

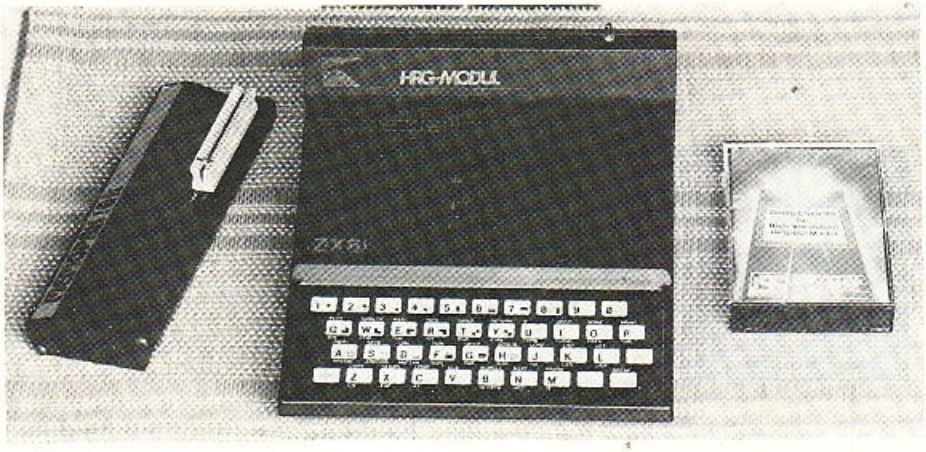
Mit dem HRG-Modul muß zunächst RAMTOP um z. B. 6,3 KByte herabgesetzt werden (reservieren einer einzigen Grafikseite). Damit der ZX 81 diese Seite findet, ist anschließend der Variablen V die erste Adresse des reservierten Speicherbereichs zuzuweisen (z. B. LET V=26368). Jetzt kommt ein „Kommandostring" ZS ins Spiel. Lautet er LET Z$="STARCH", dann wird durch die folgende Anweisung RAND USR 8192 die Grafikseite im Speicher „verankert".

Um jetzt ein Pixel in der Bildmitte zu setzen sind vorbereitend die Anweisungen LET X=124 und LET Y=96 nötig (Koordinaten der Bildmitte). Dann muß man dem HRG-Modul, wieder über den Kommandostring mitteilen, daß man etwas plotten möchte: LET Z$="PLOT" erledigt dies, aber erst der erneute HRG-Aufruf RAND USR 8192 setzt endlich das Pixel an der gewünschten Stelle.

Jeder Zugriff auf eine Grafikseite verlangt in Programmen nach dem Kommandostring und dem anschließenden HRG-Aufruf (*Bild 2*). Während der HRG-Aufruf immer gleichlautend ist, gibt es für den Kommandostring 30 Befehle. Die wichtigsten bieten folgendes: inverse Bilddarstellung; drucken einer Grafikseite; rollen und scrollen von Bildteilen; aufrufen selbst definierter Grafikzeichen; ziehen einer Linie; abfeuern eines Laserstrahls; löschen von Zeichen, Punkten oder Linien; prüfen ob ein Bildpunkt gesetzt ist; einfügen von ZX-81-Zeichen; umschalten auf ZX-81-Bildspeicher; kopieren einer Grafikseite in ein Variablenfeld (zum Speichern auf Band) und zurückkopieren nach dem Laden von Band. Wegen dieser Befehle ist das HGR-Modul sehr vielseitig verwendbar, wenn man den hohen Programmieraufwand in Kauf nimmt.

Die Druckfunktion konnten wir leider nicht ausprobieren, da das Testmodell beim Anschluß des ZX-Druckers den Computer zum „Absturz" brachte. Ein Vergleichstest durch die Vertriebsfirma soll diesbezüglich jedoch geklappt haben.

Schnelligkeit darf man bei bewegter Grafik aber keinesfalls erwarten: Um nur einen einzelnen Bildpunkt von links nach rechts über den Bildschirm wandern zu lassen benötigt das Programm (Bild 2) im SLOW-Modus gut 30 s. Das Beispiel zeigt auch, daß zuweilen programmtechnische Klimmzüge nötig werden können: Hier deshalb, weil sowohl PLOT als auch UNPLOT nur dieselbe Übergabe-Variable X akzeptieren. -ll



① **ZX 81 mit HRG-Modul:** Hinter dem Grafik-Modul muß eine Speichererweiterung von mindestens 16 KByte aufgesteckt werden Foto: -ll

```
 10 LET V=26368
 20 LET Z$="STARCH"
 30 RAND USR 8192
 40 LET Y=100
 50 FOR X=1 TO 247
 60 LET Z$="PLOT"
 70 RAND USR 8192
 75 LET X=X-1
 80 LET Z$="UNPLOT"
 90 RAND USR 8192
 95 LET X=X+1
100 NEXT X
```

② **Programmbeispiel:** Dieses Programm bewegt einen einzelnen Bildpunkt vom linken zum rechten Rand des Bildschirms

Computer lernen Sprechen:

Sprachschatz

Beliebige Worte, ja sogar vollständige Sätze kann der hier vorgestellte Sprach-Synthesizer sprechen, wenn man aus seinem Sprachschatz Elementarlaute sinnvoll aneinanderreiht.

Die „Phonem-Sprachsynthese" ist zwar anderen Verfahren in der Sprachqualität unterlegen. Vielseitig und preislich attraktiv, ist sie jedoch für experimentierfreudige Hobby-Elektroniker genau das richtige.

Phoneme sind Elementarlaute der Sprache, die allerdings nicht mit einem Buchstaben übereinstimmen müssen; sie sind vergleichbar mit den Lautschrift-Zeichen in Fremdsprachen-Wörterbüchern. Der verwendete Sprachsynthese-Prozessor SP 0256-AL2 kann z. B. 64 Phoneme erzeugen, aus denen dann der zu sprechende Text zusammengesetzt wird.

Der Synthese-Prozessor ist das Herz der Schaltung

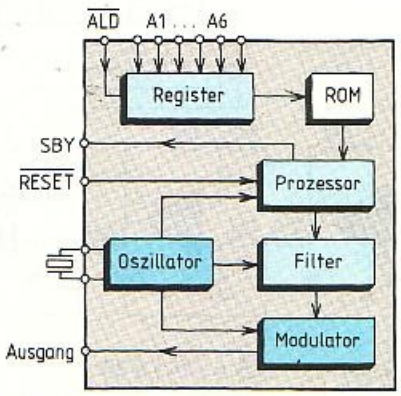
Bild 1 zeigt die Blockschaltung des Synthese-Prozessors. Damit man von den 64 Phonemen jedes einzelne auswählen kann, sind sechs Dateneingänge (A1 bis A6) notwendig ($2^6 = 64$). Dann gibt es einen Datenübernahme-Eingang ($\overline{ALD}$) sowie einen Rückmelde-Ausgang (SBY). Alle diese Anschlüsse sind mit dem Datenbus des Computers verbunden. Das gilt auch für den $\overline{RESET}$-Eingang.

Querstriche über den Pin-Bezeichnungen bedeuten, daß die zugeordnete Funktion durch L-Pegel ausgeführt wird. L-Pegel am $\overline{RESET}$-Eingang bringt das IC in einen definierten Ausgangszustand.

Vor der eigentlichen Sprachausgabe ist stets ein Rücksetzimpuls notwendig. Die Ausgabe der Phoneme läuft dann zehn- bis zwölfmal pro Sekunde nach folgendem Schema ab: Die vom Computer bereitgestellten Daten (6 Bit) für die Auswahl eines Phonems werden mit einem L-Impuls am $\overline{ALD}$-Eingang in einem internen Register zwischengespeichert. Anhand dieser Daten ermittelt der eingebaute Prozessor programmgesteuert Parameter, die ein 12poliges Digitalfilter steuern. Dieses Filter bildet den menschlichen Sprachtrakt nach. Es hat die Aufgabe, aus der Grundfrequenz des externen Quarzes das typische Frequenzgemisch des gerade ausgewählten Phonems herauszufiltern.

Das Ergebnis liegt dann noch digital vor, codiert in einem 7-Bit-Datenwort. Dieses Datenwort steuert einen Pulsbreiten-Modulator. An seinem Ausgang steht das Phonem in der erforderlichen seriellen Form zur Verfügung. Ein externes Filter formt daraus schließlich das NF-Signal.

Während der Ausgabe eines Phonems kann das IC keine neuen Daten aufnehmen. Der Synthese-Prozessor gibt deshalb am Pin SBY solange L-Pegel aus, bis er mit der Ausgabe eines Phonems fertig ist. Mit diesem Signal läßt sich die Datenübertragungsgeschwindigkeit vom Computer zum Synthese-Baustein synchronisieren. Eine feste Übertragungsrate ist nicht möglich, da die Ausgabezeiten der Phoneme untereinander verschieden sind.



① **Blockschaltung des SP 0256-AL2:** Die Pegelverteilung an A1...A6 bestimmt, welches Phonem (im ROM gespeichert) pulsdauermoduliert ausgegeben wird

Grundsätzlich ist die Schaltung des Sprachsynthesizers an alle Computer anschließbar, wenn eine Hardware-Schnittstelle bereitsteht, die folgenden Eigenschaften genügt: 16 Bit Adreßbus; 8 Bit Datenbus (bidirektional); getrennte Schreib- und Lesesignale für den Speicherzugriff ($\overline{WR}$ und $\overline{RD}$). Nur fallweise sind Leitungen notwendig, um Speicher abzuschalten ($\overline{RAMCS}$ und $\overline{ROMCS}$).

Bei Computern, die als Zentraleinheit den Mikroprozessor Z 80 verwenden, sind diese Voraussetzungen z. B. erfüllt. Getestet wurde die Schaltung in Verbindung mit dem ZX 81, für ihn ist sie deshalb ohne Veränderungen verwendbar.

Adressiert wird der Sprachsynthesizer wie eine Hauptspeicherzelle. Von Basic aus läßt er sich also mit PEEK und POKE steuern, so daß Kenntnisse in Maschinensprache nicht nötig sind.

Das Freigabe-Signal hat einen langen Weg

Ist beim Computer unter der verwendeten Adresse tatsächlich Speicher (RAM) vorhanden, so muß dieser für die Dauer des Zugriffs abgeschaltet werden – daher die Speicher-Abschaltleitungen. Beim ZX 81 schaltet man besser auch gleich das ROM mit ab, da dieses mehrfach im Speicherbereich auftritt; der Grund hierfür ist die unvollständige Decodierung der Adreßleitungen im ZX 81.

Wählt man für die Adresse den Wert 65535, so ist die Decodierung besonders einfach, denn dann führen alle 16 Adreßleitungen H-Pegel. Das Schaltbild (*Bild 2*) zeigt die Decodierung:

Zunächst werden die Adreßsignale A0 bis A15 mit den Treibern IC 1 und IC 2 verstärkt, um die Belastung des Adreßbusses gering zu halten. Von zwei Achtfach-NAND-Gattern (IC 3 und IC 4) werden diese Signale zusammengefaßt. An den Ausgängen dieser beiden Gatter liegt nur dann L-Pegel, wenn alle Adreßsignale H-Pegel haben.

Nun müssen wiederum die beiden Ausgänge von IC 3 und IC 4 zusammengefaßt werden, und zwar so, daß der resultierende Ausgang immer dann L-Pegel führt, wenn Adresse 65535 am Adreßbus anliegt. Das erreicht man durch drei NAND-Gatter (IC 5), von denen zwei zum Invertieren dienen und eines zum Zusammenfassen. An Pin 11

von IC 5 haben wir dann ein Freigabe-Signal (Enable-Signal) für die Schaltung, das allerdings noch mit anderen Signalen verknüpft werden muß.

Alles wird über den Datenbus abgewickelt

Um den Sprachsynthese-Prozessor (IC 9) mit Datenworten zu steuern, genügt eine Wortbreite von 8 Bit (1 Byte). Sechs Bit davon dienen zur Auswahl eines der 64 Phoneme, ein Bit signalisiert dem $\overline{ALD}$-Eingang, daß Daten anliegen, und das achte Bit dient zum Rücksetzen ($\overline{RESET}$).

Die Datensignale D0 bis D7 werden ebenfalls verstärkt (IC 6) und anschließend auf die Eingänge eines 8-Bit-Speichers (IC 8) gelegt. Dieses IC ist notwendig, weil die für den Sprachsynthese-Prozessor vorgesehenen Informationen nur kurzzeitig auf dem Datenbus liegen. Das Zwischenspeichern der Daten ist jedoch an zwei Bedingungen geknüpft: die Adresse muß stimmen (Enable-Signal an Pin 1 auf L) und der Computer muß einen Schreib-Impuls ($\overline{WR}$) ausgeben. Der Schreib-Impuls wird über einen Inverter (IC 7) auf den Datenübernahme-Eingang (Pin 11) von IC 8 gelegt.

Das Enable-Signal liegt nicht nur an IC 8, sondern auch am Freigabe-Eingang (Pin 1) von IC 7. Hiermit wird gewährleistet, daß nur dann Daten transportiert oder Steuersignale an IC 9 weitergereicht werden, wenn die richtige Adresse auf dem Adreßbus liegt.

Es genügt aber nicht, Daten nur an den Synthese-Prozessor zu geben, denn der muß den Schaltzustand des SBY-Ausgangs an den Computer zurückmelden können (Bereitschaft zur Aufnahme neuer Daten). Dazu invertiert man zunächst das Lesesignal $\overline{RD}$ zweimal (IC 7).

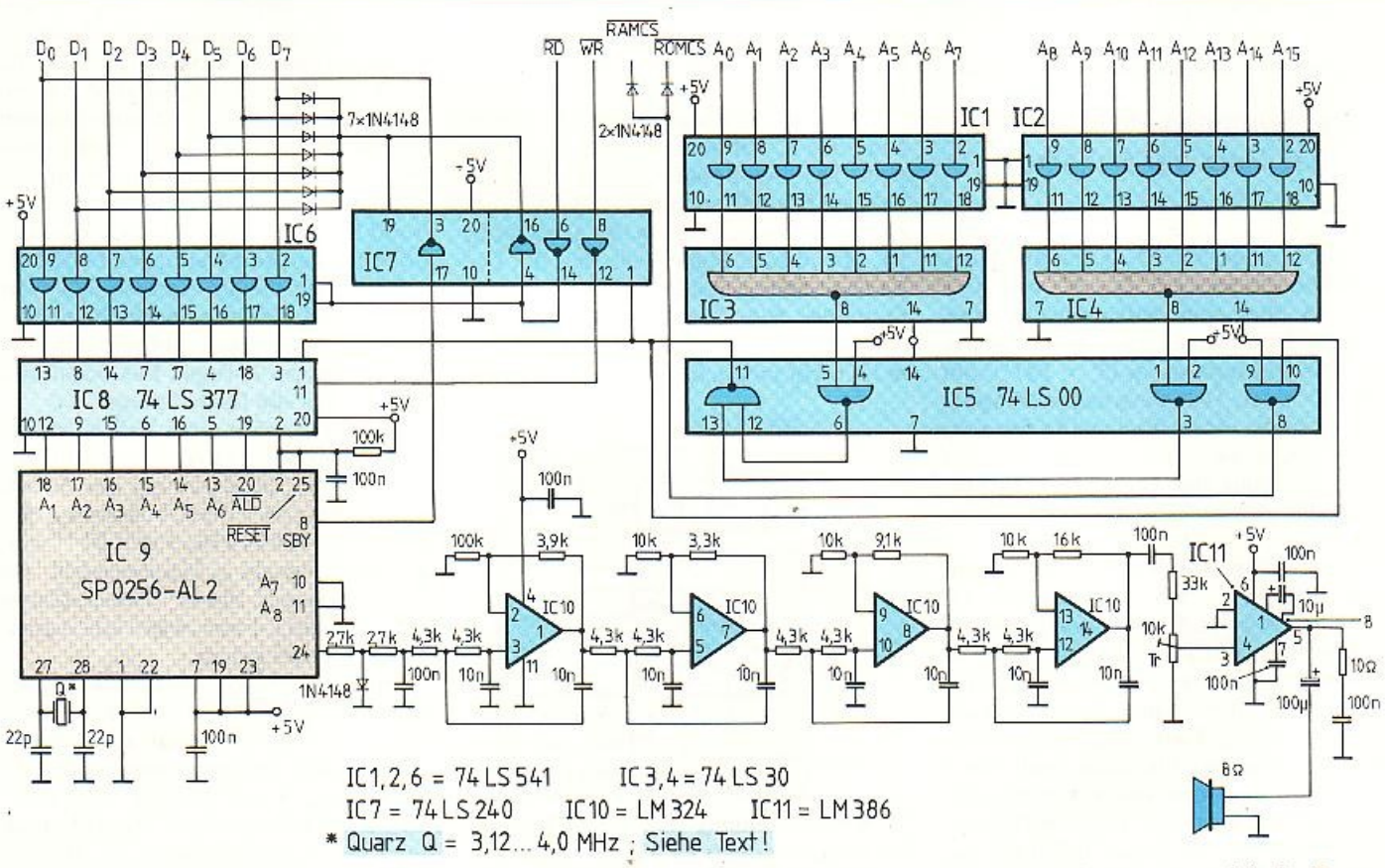
Das so verstärkte Signal aktiviert dann die andere Hälfte von IC 7 (Pin 19). Über einen davon betroffenen Inverter wird das vom Synthese-Prozessor kommende SBY-Signal auf Leitung D0 des Datenbusses gelegt. Jetzt sind nur noch die übrigen Datenleitungen auf einen definierten logischen Pegel (L) zu bringen. Hierzu wird einfach das zweifach invertierte $\overline{RD}$-Signal über je eine Diode auf die verbleibenden Datenleitungen geschaltet.

Erst wenn das Enable-Signal den Baustein IC 7 aktiviert, hat er seine Schaltfunktion, sonst ist er hochohmig. Und deshalb findet die Rückmeldung zum Rechner nur dann statt, wenn die Adresse stimmt und ein Lesesignal anliegt.

Speicher-ICs werden üblicherweise durch L-Pegel an ihren Chip-Select-Eingängen aktiviert. Wird unter Adresse 65535 also eine tatsächlich vorhandene Speicherzelle angesprochen, dann muß man diese Chip-Select-Eingänge auf H-Pegel legen. Invertieren wir das Enable-Signal mit dem letzten verfügbaren NAND-Gatter von IC 5, so steht auch dieses Signal bereit. Es ist über zwei Dioden abgreifbar, damit es nicht zu Verkopplungen und Kurzschlüssen kommt.

Der Quarz an IC 9 sollte laut Hersteller eine Grundfrequenz von 3,12 MHz haben. Es wurde aber festgestellt, daß noch Werte bis 4 MHz vertretbar sind.

Der Analogteil zur Demodulation des pulsbreitenmodulierten Signals ist relativ einfach. Die Demodulation übernimmt ein 8poliges Butterworth-Filter, das mit einem Vierfach-Operationsverstärker (IC 10) aufgebaut ist. Dieser Tief-



② **Schaltung des Sprach-Synthesizers:** Sie läßt sich an jedem Computer betreiben, der die oben eingetragenen Adreß-, Daten- und Steuersignale bereitstellt

paß läßt nur Signale mit Frequenzen unter 5 kHz durch. Nach dem Filtern folgt letztendlich ein Verstärker (IC 11).

Die Leiterplatte (*Bild 3*) ist für eine 31polige DIN-Steckerleiste ausgelegt. Mit einer dazupassenden Buchsenleiste und einem für den Computer passenden Stecker läßt sich ein Adapter bauen.

Beim Hantieren mit dem SP 0256-AL2 ist zu beachten, daß es sich hier um einen MOS-Baustein handelt. Es gelten deshalb die üblichen Vorsichtsmaßnahmen. Auf jeden Fall ist es ratsam, dieses IC – wie auch die übrigen ICs – zu sokkeln (*Bild 4*).

Der Sprach-Synthesizer ist unmittelbar an Computer anschließbar, Zusätze sind nicht nötig. Deshalb aber läßt sich die Schaltung auch vereinfachen. So dürfen z. B. IC 1, IC 2 und IC 6 entfallen, wenn die Busse bereits gepuffert sind, oder keine anderweitige Belastung durch Peripheriegeräte auftritt. Noch größer ist die Einsparung, wenn man über eine parallele I/O-Schnittstelle verfügt. Dann ist die gesamte Adreßdecodierung nicht mehr erforderlich.

Zum Verständnis der Programmierung ist ein Blick auf den zeitlichen Verlauf der Spannungen an den einzelnen Steuereingängen von IC 9 notwendig (*Bild 5*). Der erste Schritt ist das Rücksetzen des Synthese-Prozessors.

Ein Steuerprogramm regelt den Datenverkehr

Die Bits D0 bis D5 sollten dabei auf L und D6 sollte auf H liegen. Bit D7 muß dann den Schaltzustand L für mindestens 100 µs annehmen (Zeit zwischen 1 und 2). Das Programmteil zum Rücksetzen sieht demnach so aus:

```
LET ADR=65535
POKE ADR,192
POKE ADR,64
POKE ADR,192
```

Durch POKE ADR,192 werden Bit 7 und Bit 6 auf H gebracht, alle übrigen Bits nehmen L an (192 = HHLL LLLL). Anschließend bewirkt POKE ADR,64, daß nur Bit 7 seinen Wert ändert, nämlich auf L. Mit POKE ADR,192 nimmt dieses Bit wieder H-Pegel an. Damit haben wir den RESET-Impuls erzeugt, wobei die Befehlsausführungszeiten die 100-µs-Bedingung erfüllen.

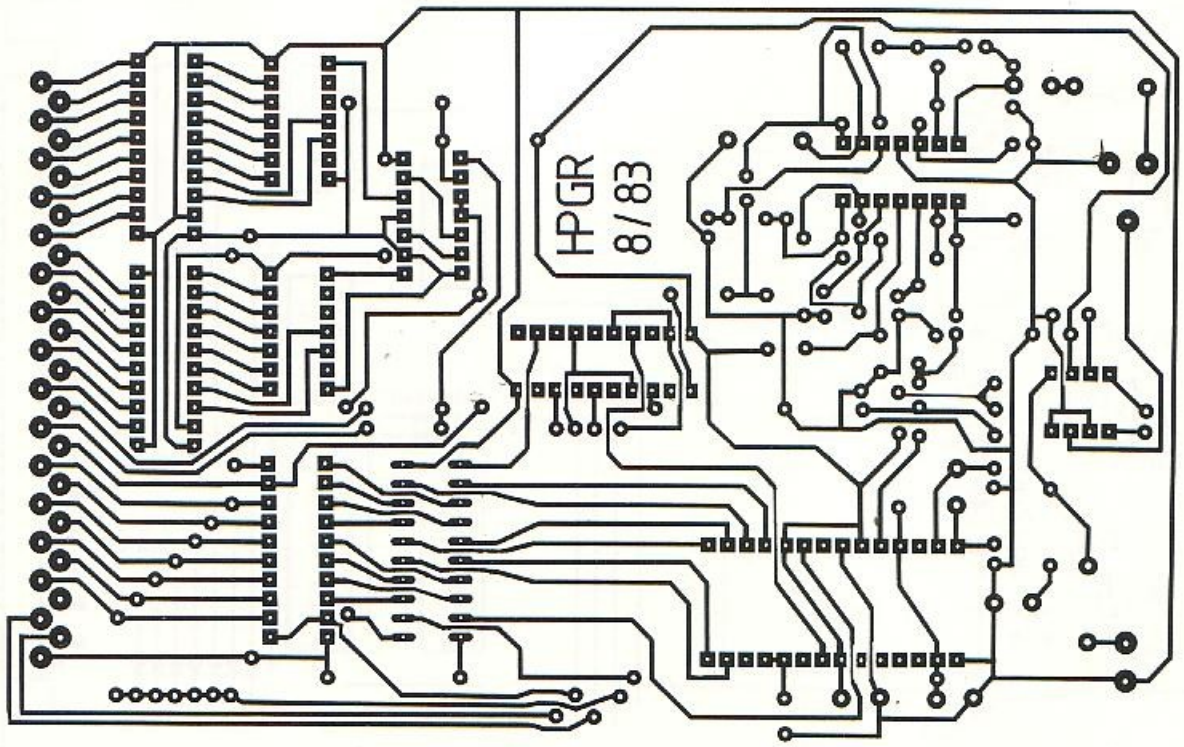
Das Rücksetzen ist nur einmal vor dem eigentlichen Programm nötig. Dieses Programm übernimmt die Ausgabe der Phoneme in der gewünschten Reihenfolge (siehe Kasten auf Seite 107):

Zuerst werden die Daten D0 bis D5 mit POKE auf den Datenbus gelegt. Diese Daten (Dezimalwert zwischen 0 und 63, nachfolgend als CODE bezeichnet) wählen ein Phonem aus. Gültig ist jetzt Zeitpunkt 3, d. h. Bit 6 und Bit 7 müssen auf jeden Fall H-Pegel haben. Anschließend werden die Daten mit einem auf D6 erzeugten $\overline{\text{ALD}}$-Impuls (Zeitpunkt 4 bis 6) vom Synthese-Prozessor übernommen.

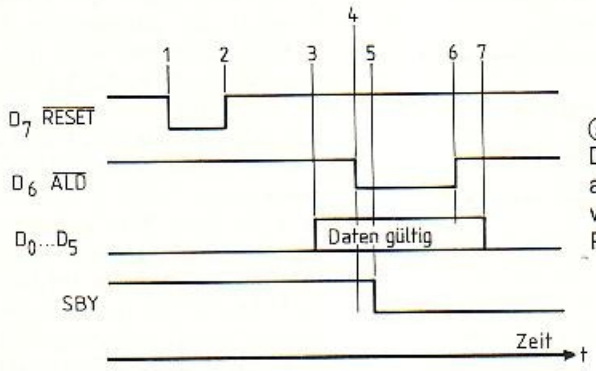
Die Zeitdauer zwischen Zeitpunkt 3 und 4 darf jedoch 450 ns nicht unterschreiten. Außerdem müssen die Daten mindestens bis zum Ende der Übernahmephase (Zeitpunkt 6) ihren Wert behalten. Hier die Lösung:

```
POKE ADR,CODE+192
POKE ADR,CODE+128
POKE ADR,CODE+192
```

Mit CODE wird ständig der Dezimalcode des gewünschten Phonems auf den Datenleitungen D0 bis D5 ausgegeben. Durch die Addition von 192 werden zwangsweise die Bits bzw. Datenleitungen D6 und D7 auf H-Pegel gebracht (Zeitpunkt 3). Die sechs Datenbits bleiben davon unbeeinflußt!



③ **Platinenlayout:** Ein Liefernachweis für die fertig geätzte Platine ist bei der Redaktion abrufbar



⑤ **Zeitdiagramm:** Die Schaltimpulse auf D6 und D7 werden per Programm erzeugt

```
  1 REM E?¡wL-$O▪▪▪▪000
  2 FOR X=16514 TO 16525
  3 PRINT AT 1,1;PEEK X,AT 1,10;
X
  4 INPUT P$
  5 IF P$="" THEN NEXT X
  6 POKE X,VAL P$
  7 NEXT X
  8 STOP
 10 LET A=65535
 30 POKE A,192
 40 POKE A,54
 50 POKE A,192
 60 FOR X=16514 TO 16525
 70 LET C=PEEK X
 80 POKE A,192+C
 90 POKE A,128+C
100 POKE A,192+C
110 IF PEEK A=1 THEN GOTO 110
120 NEXT X
130 GOTO 60
```

⑥ **Steuerprogramm:** Die Phonemcodes werden in einem REM-Kommentar untergebracht

Das Addieren von 128 hat nur eine Auswirkung: Bit 6 ändert damit seinen Wert auf L (Zeitpunkt 4). Erneutes Addieren von 192 setzt dieses Bit wieder auf H (Zeitpunkt 6), womit die Datenübernahme abgeschlossen ist.

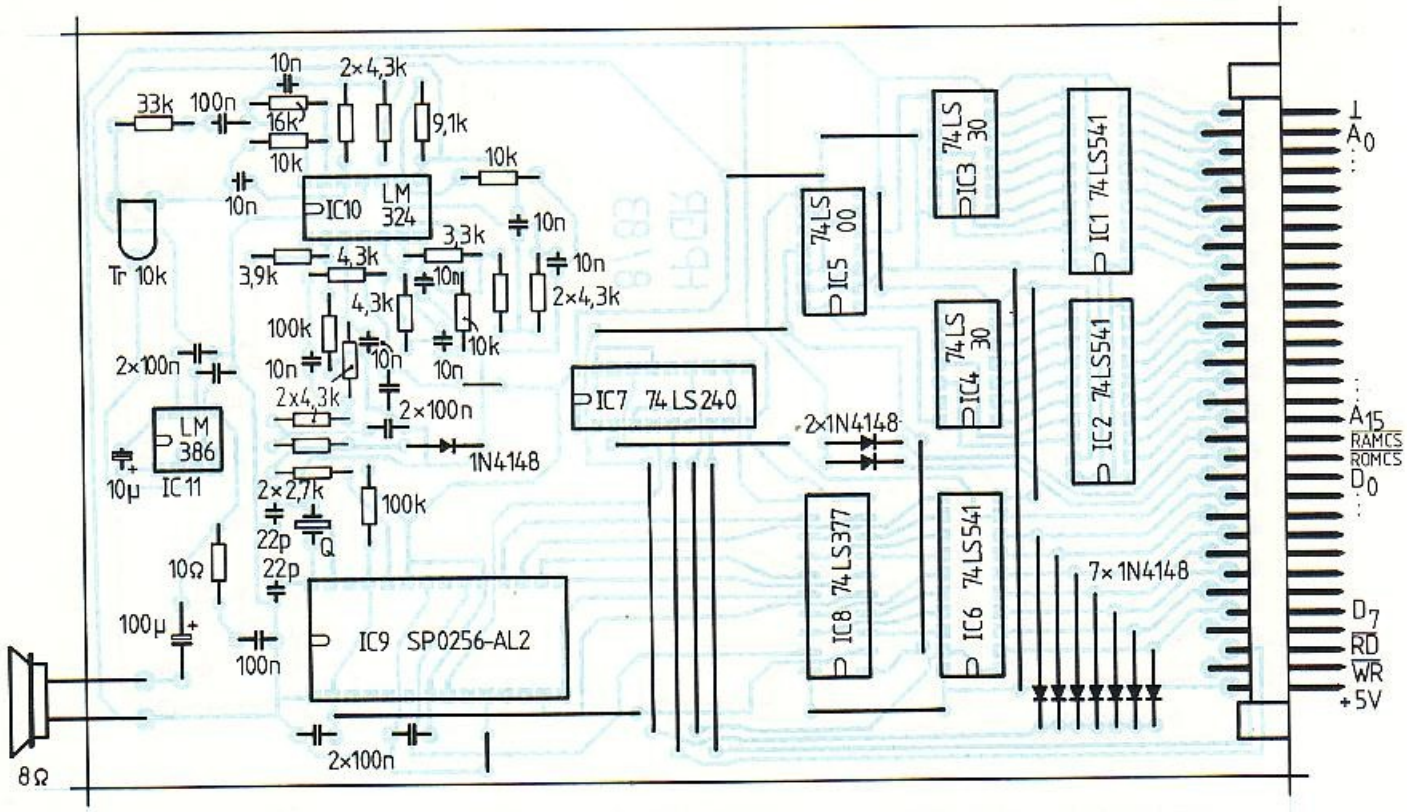
Kurz nach Übernahme der Daten in das Pufferregister des Synthese-Prozessors nimmt dessen Ausgang SBY den Wert L an. Das IC meldet sich damit sozusagen ab und beginnt mit der Phonemausgabe. Erst wenn es damit fertig ist, nimmt SBY wieder den Wert H an. Es muß also vom Programm noch nachgefragt werden, ob ein erneuter Datentransport Aussicht auf Erfolg hat. Ist das nicht der Fall (D0 = 1 bzw. H), muß der steuernde Computer warten. Folgende Programmzeile löst dieses Problem:

100 IF PEEK ADR=1 THEN GOTO 100

Führt man den Datentransport und die Rückfrage immer abwechselnd aus, so werden der Reihe nach die Codes der benötigten Phoneme an den Synthese-Prozessor weitergegeben, der sie zu Sprache umformt.

Einen Vorschlag für das komplette Steuerprogramm (ZX 81) zeigt Bild 6. Der Programmblock bis Zeile 8 dient lediglich zur Eingabe der dezimalen Phonemcodes. Diese Codes (bzw. die entsprechenden ZX-81-Zeichen) werden, mit POKE im REM-Kommentar (Zeile 1) beginnend, ab Adresse 16514 untergebracht. Der REM-Kommentar muß deshalb vor dem Programmstart (RUN) mindestens ebenso viele Nullen haben (als Platzhalter für die Phonemcodes) wie Phonemcodes eingegeben werden.

Bild 6 zeigt die bereits durch POKE veränderte REM-Zeile nach Eingabe der Codes für das Wort „Computer". Zum Nachvollziehen ist die REM-Zeile zunächst mit mindestens zwölf Nullen zu füllen. Nach dem Programmstart sind



④ **Bestückungsplan:** Sämtlichen ICs sollte man Sockel spendieren

dann folgende Phonemcodes einzutippen: 42, 15, 16, 9, 49, 22, 13, 52, 4, 4, 4, 4. Zum Experimentieren kann man der FOR-NEXT-Schleife (Zeile 2) auch eine höhere Endadresse zuordnen (höchstens aber 16514 zuzüglich Zahl der Nullen im REM-Kommentar).

Die tatsächliche Endadresse (sie wird in Zeile 60 benötigt) läßt sich dann vor Eingabe des letzten Phonemcodes am Bildschirm (rechts) ablesen. Ebenso kann man nach erneutem Programmstart mit RUN prüfen, ob die Eingaben richtig waren, denn die Speicherzelleninhalte werden am Bildschirm (links) angezeigt (Zeile 3). Zum „Durchtasten" ist dann nur fortwährend NEWLINE zu betätigen (Zeile 5) – die Speicherzelleninhalte werden dabei nicht verändert.

Und jetzt wird's spannend: Nach dem Start des Steuerprogramms (RUN 10) muß aus dem Lautsprecher das Wort „Computer", getrennt durch eine kurze Pause, immer wieder ertönen (Lautstärke am Trimmer einstellen). Der ZX 81 sollte im FAST-Modus arbeiten, sonst ist die Sprechgeschwindigkeit zu niedrig.

Das Steuerprogramm enthält die bereits besprochenen Programmteile. Mit der FOR-NEXT-Schleife in Zeile 60 wird nur dafür gesorgt, daß die im REM-Kommentar untergebrachten Phonemcodes des Wortes der Reihe nach an den Synthese-Prozessor übergeben werden. Die letzten vier Phonemcodes bewirken die Pause. Anschließend beginnt die Wortausgabe von neuem (Zeile 130).

Noch ein Tip zum Betrieb der Schaltung am ZX 81 mit 16 KByte RAM. Bei den Memotech-RAMs ist die Adreßleitung A 14 nicht durchgeschleift. Ein Steckadapter, der das RAM als letztes aufnimmt, löst dieses Problem. Man kann aber auch auf der Sprachsynthese-Platine A 14 zwangsweise über einen 4,7-kΩ-Widerstand mit H-Pegel versorgen (dann aber Leitung zum Stecker auftrennen), damit die Adressierung wieder stimmt. In diesem Fall macht sich die Speicherabschaltung bezahlt, denn sie verhindert dann eine Doppeladressierung.

Hans Peter Gramatke

Der Phonemvorrat des SP 0256-AL2

Erste und letzte Warnung: Der Sprachsynthesizer ist sowohl hard- als auch softwaremäßig eine harte Nuß, die zu knacken eine gehörige Portion Erfahrung fordert. Ertönt aber erst einmal das Testwort „Computer", dann beginnt eine aufregende Experimentierphase. Bevor der Computer nämlich das plappert, was er soll, gilt es, mit detektivischem Spürsinn die richtigen Phoneme aus der Tabelle (unten) auszuwählen.

Wie die Musterworte zeigen, ist der Phonemsatz aus dem Amerikanischen entlehnt – deutsch wird deshalb immer mit einem unverkennbaren Akzent gesprochen. Die Verständlichkeit der synthetischen Stimme hängt maßgebend von einer geschickten Phonem-Zusammenstellung ab. Hohen Ansprüchen genügt sie nicht: Die Stimme ist gewöhnungsbedürftig – dann aber durchaus zu verstehen. Tests in der Redaktion zeigten dabei ein verblüffendes Ergebnis: Ein Mitarbeiter mit vermindertem Hörvermögen verstand den Sprachsynthesizer auf Anhieb. Ihm half offenbar der zur Gewohnheit gewordene Zwang, undeutlich gehörte Worte durch Interpretation dennoch zu erkennen.

Zum Aufspüren der passenden Phoneme ist es hilfreich, sich das jeweilige Wort laut vorzusagen. Dann kann man vergleichen, welches Phonem am besten zu den einzelnen Lauten paßt: KK1-AX-MM-PP-YY1-UW1-TT2-ER2 wäre z. B. ein Vorschlag für das Wort „Computer". Die entsprechenden Dezimalcodes dazu werden dann einfach der Reihe nach eingetippt (Eingabeprogramm Bild 6).

Hier noch einige Hilfen: PA1 oder PA2 sollte man vor die Phoneme BB, DD, GG und JH setzen, PA3 vor PP, TT, KK und CH sowie zwischen Worte. Stehen die Laute YR, IY, EY, EH, XR, AE, ER, AX, AW und UW am Wortanfang oder in der Wortmitte, so ist der Nasal-Laut NN1 davorzusetzen. Dagegen ist NN2 vor die Laute UH, OW, OY, OR, AR und AA zu setzen, wenn diese am Schluß eines Wortes stehen. Hierbei darf man nach Herzenslust Experimentieren, z. B., um die Phonemfolge für eine Punkte-Ansage zu bekommen. Denn selbstverständlich läßt sich das Syntheseprogramm auch als Unterprogramm aufrufen, um irgendwelche Kommentare, Ergebnisse oder Aufforderungen am Bildschirm verbal zu bekräftigen.

Ein Liefernachweis für den Synthese-Prozessor ist bei der Redaktion abrufbar. Dem IC liegt eine englischsprachige Dokumentation bei mit weiteren Hinweisen, die die Phonemauswahl erleichtern.

Code				Phonem				Musterwort				Dauer in ms			
00	PA1	Pause	10	16	MM	**Milk**	180	32	AW	**Out**	250	48	WH	**Whig**	150
01	PA2	Pause	30	17	TT1	**Part**	80	33	DD2	**Do**	80	49	YY1	**Yes**	90
02	PA3	Pause	50	18	DH1	**They**	140	34	GG3	**Wig**	120	50	CH	**Church**	150
03	PA4	Pause	100	19	IY	**See**	170	35	VV	**Vest**	130	51	ER1	**Fir**	110
04	PA5	Pause	200	20	EY	**Beige**	200	36	GG1	**Guest**	80	52	ER2	**Fir**	210
05	OY	**Boy**	290	21	DD1	**Could**	50	37	SH	**Ship**	120	53	OW	**Beau**	170
06	AY	**Sky**	170	22	UW1	**To**	60	38	ZH	**Azure**	130	54	DH2	**They**	180
07	EH	**End**	50	23	AO	**Aught**	70	39	RR2	**Brain**	80	55	SS	**Vest**	60
08	KK3	**Comb**	80	24	AA	**Hot**	60	40	FF	**Food**	110	56	NN2	**No**	140
09	PP	**Pow**	150	25	YY2	**Yes**	130	41	KK2	**Sky**	140	57	HH2	**Hoe**	130
10	JH	**Dodge**	100	26	AE	**Hat**	80	42	KK1	**Can't**	120	58	OR	**Store**	240
11	NN1	**Thin**	170	27	HH1	**He**	90	43	ZZ	**Zoo**	150	59	AR	**Alarm**	200
12	IH	**Sit**	50	28	BB1	**Busy**	40	44	NG	**Anchor**	200	60	YR	**Clear**	250
13	TT2	**To**	100	29	TH	**Thin**	130	45	LL	**Lake**	80	61	GG2	**Got**	80
14	RR1	**Rural**	130	30	UH	**Book**	70	46	WW	**Wool**	140	62	EL	**Saddle**	140
15	AX	**Succeed**	50	31	UW2	**Food**	170	47	XR	**Repair**	250	63	BB2	**Busy**	60

ZX-81-Hardwaretip:

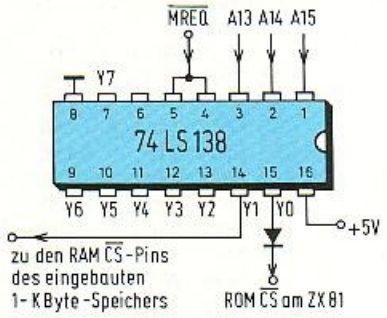
Wiederbelebung

Vollständige Adreßdecodierung des ROMs

Und noch ein Knüller für ZX-81-Fans: Durch „Wiederbelebung" des internen 1-KByte-RAMs wird ein Speicher gewonnen, der nach einem „Absturz" des Computers nicht zwangsweise gelöscht wird. Eine zweite Lösung schafft sogar bis zu 8 KByte sicheren Speicherplatz.

Darüber haben sich insbesondere Freunde der Maschinensprache beim ZX 81 schon oft geärgert: Steigt der Computer aus irgendeinem Grund mitten im Programm aus, dann bleibt einem keine andere Wahl, als resignierend den Netzstecker zu ziehen und mit der Programmiererei wieder von vorne zu beginnen.

Hoffnung auf einen gnädiger gestimmten ZX 81 keimt, wenn man entdeckt, daß die Z-80-CPU einen Reset-Eingang hat (Pin 26). Legt eine Taste diesen Eingang kurz auf L-Pegel, sollte der Computer eigentlich wieder Tritt fassen und ein Wiedersehen mit dem Programm möglich sein. Leider trügt die Hoffnung, denn nach einem L-Impuls an Pin 26 der CPU feiert das Betriebssystem des Computers erst einmal fröhliche Urstände – und führt den mit Adresse 0 beginnenden Programmteil „Urstart" aus.



① **Hürdensprung nach oben:** Diese Schaltung nutzt den stillgelegten 1-KByte-Grundspeicher. Steigt der ZX 81 aus, bleibt der Speicherinhalt oberhalb 32 KByte erhalten

Dabei wird zuerst mit einem out-Befehl der NMI-Generator ausgeschaltet (bewirkt u. a. Einschalten des FAST-Modus) und dann werden, ausgehend von Adresse 32 768 bis herunter zur Adresse 16 383, alle dazwischenliegenden Speicherzellen mit dem Wert 02 geladen. Anschließend prüft die CPU in umgekehrter Richtung, ob die Speicherzellen – nach zweimaligem Decrementieren des Inhalts (je 1 abziehen) – den Wert 0 haben. Die erste Speicherzelle die dann einen von 0 abweichenden Wert hat wird als Ende des RAM-Speichers erkannt und die zugehörige Adresse in der Systemvariablen RAMTOP abgelegt.

Das gleiche macht der ZX 81 jedesmal nach dem Einschalten, er weiß danach stets, wieviel Bytes RAM-Speicher ihm zur Verfügung stehen. Nur bringt uns das keinen Schritt weiter, weil durch den „Speichertest" ein im RAM stehendes Programm zwangsläufig überschrieben wird und dabei verlorengeht.

Das ROM ist der Übeltäter

Das ROM des ZX 81 umfaßt 8 KByte und liegt im Adreßbereich zwischen 0 und 8191. Um jede einzelne Speicherzelle des ROMs ansprechen zu können, sind insgesamt 13 Adreßleitungen nötig ($2^{13} = 8192$).

Woher aber „weiß" das ROM, daß es im Adreßbereich zwischen 0 und 8191 liegt? Das ROM selbst kann das nicht erkennen, jedoch der Sinclair-Logik-Chip (SLC). Er enthält u. a. einen „Adreßdecoder" für das ROM. Der wiederum bewirkt, daß bei Adressen zwischen 0 und 8191 am Ausgang ROM $\overline{CS}$ des SLC L-Pegel herrscht. Und genau mit diesem Pegel wird auch das ROM aktiviert (Pin 20, ROM $\overline{CS}$-Eingang), d. h. es legt die zur jeweiligen Adresse gehörenden Daten auf den Datenbus. H-Pegel an Pin 20 sperrt das ROM. Für den ZX 81 ist es dann scheinbar nicht mehr vorhanden. Das ist z. B. der Fall, wenn ein RAM-Baustein adressiert wird.

Der ROM $\overline{CS}$-Ausgang am SLC und der entsprechende Eingang am ROM sind nicht direkt, sondern über einen Widerstand miteinander verbunden. Deshalb können wir – wie später gezeigt wird – das ROM mit externen Signalen an Pin 20 (bzw. an der Schnittstelle) sperren oder aktivieren, ohne daß der SLC Schaden nimmt. Zunächst ist es jedoch wichtiger zu wissen, daß der Adreßdecoder im SLC nur die für das ROM „interessanten" 13 Adreßleitungen A0 bis A12 abfragt.

Die CPU soll aber nicht nur das ROM, sondern noch jede Menge RAM adressieren können. Sie hat deshalb 16 Adreßleitungen, ausreichend für einen Adreßbereich von 65 536 Byte ($2^{16} = 65\,536$). Solange die CPU jetzt nur auf den 13 Adreßleitungen A0 bis A12 Adressen ausgibt ist alles klar: Dann wird das ROM angesprochen.

Was aber geschieht, wenn die CPU eine höhere Adresse auf den Adreßbus legt z. B. 8200? Unter dieser Adresse müßte sich schon eine RAM-Speicherzelle ansprechen lassen. Nein, denn jetzt treibt das ROM sein Unwesen.

Da der Adreßdecoder im SLC nur die Adreßleitungen A0 bis A12 decodiert, erkennt er nicht, daß bei Adresse 8200 auch auf der Leitung A13 ein Signal liegt (H-Pegel). Allein dieses Signal ermöglicht aber erst eine Unterscheidung zwischen den Adressen 0 bis 8191 und 8192 bis 16 383. Ein Beispiel: Bei Adresse 0 führen die Adreßleitungen A0 bis A13 L-Pegel. Bei Adresse 8192 haben die Leitungen A0 bis A12 ebenfalls wieder L-Pegel, aber A13 führt H-Pegel. Unter der Adresse 8200 wird also ungewollt zusätzlich die Adresse 8 im ROM angesprochen (8200 – 8192 = 8)! Schuld daran ist die „unvollständige Adreßdecodierung" durch den SLC.

Gäbe es unter der Adresse 8200 noch eine RAM-Speicherzelle käme es zu einer verbotenen Doppeladressierung. Insgesamt taucht das störende ROM-Doppel sogar dreimal im Adreßbereich auf: ab 8192, ab 32 768 und ab 40 960. Damit

haben auch die beiden Grenzwerte beim Speichertest ihren Sinn. Sie stecken einen Adreßbereich ab, in dem das ROM-Doppel nicht stören kann.

Jetzt können wir den ZX 81 mit seinen eigenen Waffen schlagen. Der Speichertest zerstört nämlich nur Programme im Adreßbereich zwischen 16 KByte und 32 KByte. RAM-Speicher, der darüber oder darunterliegt, bleibt unbehelligt. Es muß nur gelingen, aus diesen angrenzenden Speicherbereichen das ROM-Doppel zu verbannen. Das ROM bietet uns dazu den Eingang ROM $\overline{CS}$ an. Wir wissen: H-Pegel an diesem Eingang verhindert, daß sich das ROM bemerkbar machen kann.

Wiederbelebung des 1-KByte-RAMs

Wird der ZX 81 mit einer Speichererweiterung betrieben, dann ist das in ihm eingebaute 1-KByte-RAM (zwei ICs 2114 oder ein 4118) automatisch abgeschaltet. Bei einer 16-KByte-Speichererweiterung können wir dieses RAM gemäß *Bild 1* in einem verschobenen Adreßbereich jedoch wieder in Betrieb nehmen, wobei die Speicher-ICs sogar ihren Platz behalten dürfen!

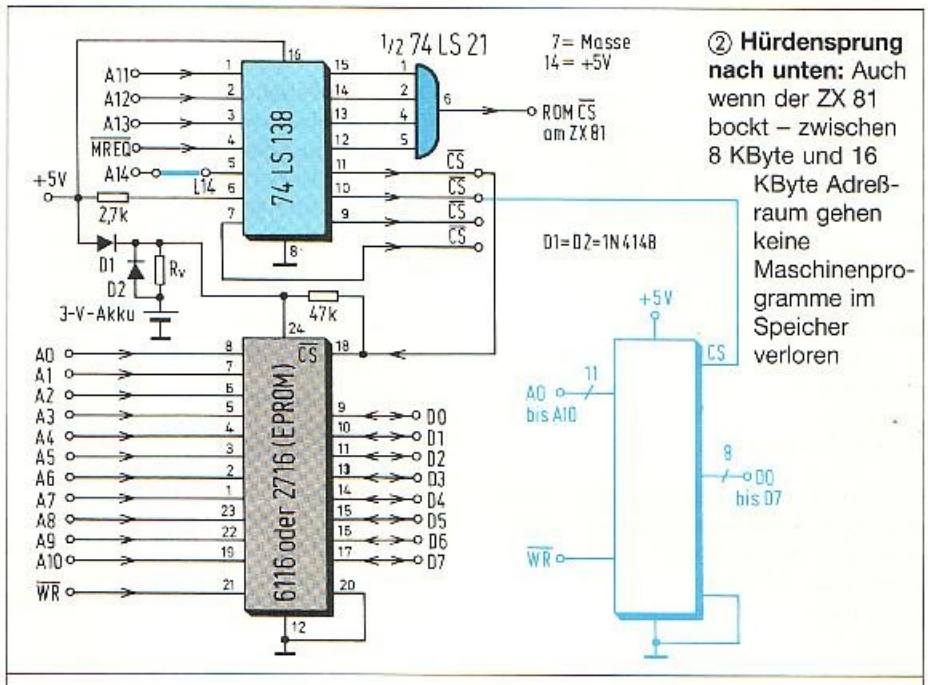
Mit dem Adreßdecoder 74LS138 wird das ROM nachträglich vollständig decodiert bzw. das 1-KByte-RAM freigegeben (Schaltung und Wahrheitstabelle des 74LS138 siehe Heft 12/1983, Seite 73, oder FUNKSCHAU-Sonderheft Nr. 14, Seite 49).

Mit dem Signal $\overline{MREQ}$ wird der Decoder selbst freigegeben, d. h., solange die CPU keinen Speicher anspricht ($\overline{MREQ}$ = H) bleiben alle Ausgänge (Y0 bis Y7) auf H-Pegel. Da sowohl ROM als auch RAM mit L-Pegel an den $\overline{CS}$-Eingängen aktiviert werden, sind somit beide gesperrt. Bei einem Speicherzugriff ($\overline{MREQ}$ = L) mit einer Adresse unter 8192 (A13 bis A15 auf L) führt allein Ausgang Y0 des Adreßdecoders L-Pegel, der, an Pin 20 des ROMs (ROM $\overline{CS}$) gelegt, dieses aktiviert. Bei Adressen über 8191 führt Y0 immer H-Pegel und sperrt damit das ROM. So liegt z. B. im Adreßbereich zwischen 8192 und 16 383 (A13 = H, A14 und A15 = L) allein Y4 auf L-Pegel.

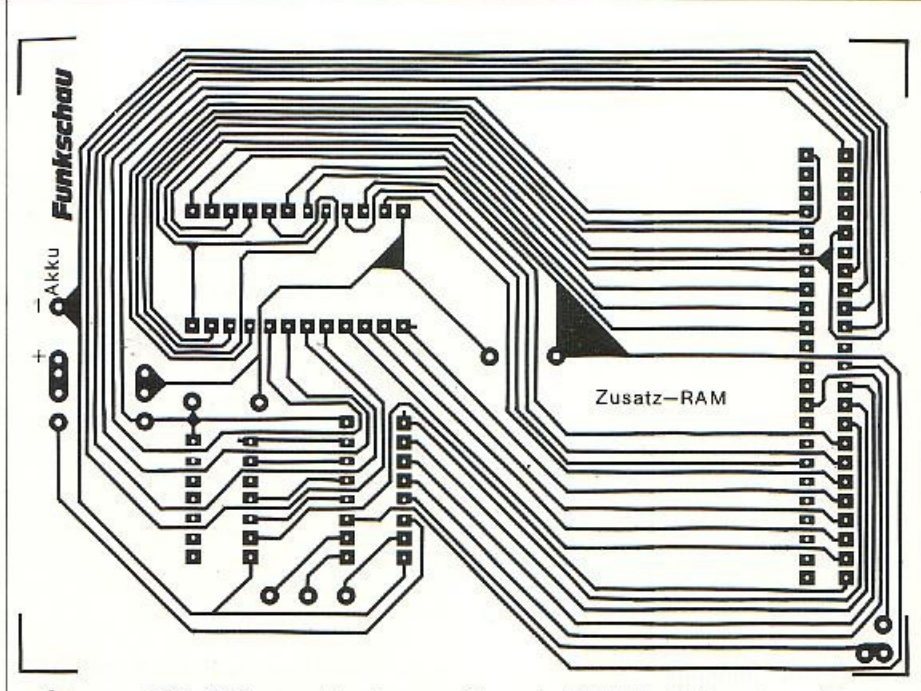
Der uns interessierende Adreßbereich ab 32 768 (bis 40 959) wird angesprochen, wenn A13 und A14 auf L-Pegel liegen und A15 H-Pegel führt. Dann nimmt Ausgang Y1 des Adreßdecoders L-Pegel an.

Soll nun der 1-KByte-Grundspeicher in diesem Adreßbereich wieder in Betrieb genommen werden, dann genügt es, Pin 8 der beiden 2114-ICs hochzubiegen (bzw. Pin 18 und 20 des 4118), und mit Pin 14 (Ausgang Y1) des Adreßdecoders zu verbinden. Das Signal $\overline{MREQ}$ kann man an Pin 19 der CPU abgreifen, die Adreßsignale A13 bis A15 an den Katoden der Tastatur-Dioden D5, D7 und D8. Der Reset-Taster sollte Pin 26 der CPU über einen 100-Ω-Widerstand auf Masse legen.

In den 1-KByte-Zusatzspeicher lassen sich jetzt mit POKE (Adresse zwischen 32 768 und 33 792) Maschinenprogramme unterbringen. Sollte der ZX 81 dann einmal aussteigen, genügt ein Druck auf die Reset-Taste um das gespeicherte Pro-



② **Hürdensprung nach unten:** Auch wenn der ZX 81 bockt – zwischen 8 KByte und 16 KByte Adreßraum gehen keine Maschinenprogramme im Speicher verloren



③ **Layout:** Die Platine ragt hoch genug über ein RAM-Modul hinaus, um bis zu vier 6116-ICs stapeln zu können

gramm wieder zu erreichen. Durch Hochsetzen von RAMTOP (POKE 16389, 132 und anschließend NEW) wird der Speicher auch für das Betriebssystem des ZX 81 erreichbar.

Gepuffert in den Keller gehen

Die Schaltung nach *Bild 2* decodiert den Adreßbereich zwischen 0 und 16 383. Sie kann also auch mit 32-KByte- und 64-KByte-RAMs genutzt werden, wobei aber nur 16 KByte tatsächlich verwendbar sind. Die geänderte Ansteuerung des Adreßdecoders bewirkt, daß seine Ausgänge jetzt in 2-KByte-Schritten L-Pegel annehmen (bei der anderen Lösung sind es 8-KByte-Schritte). Das erfordert zwar ein Zusammenfassen von vier Ausgängen mit einem UND-Gatter, um ROM $\overline{CS}$ zu gewinnen (4 × 2 KByte = 8 KByte), ermöglicht aber im Adreßbereich von 8192 bis 16 383 das Freigeben von maximal vier RAM-ICs mit je 2 KByte Speicherkapazität (Typ: 6116).

Werden diese ICs batteriegepuffert, hat man sogar einen nichtflüchtigen Speicher. Der ist freilich vom Betriebssystem des ZX 81 und damit für Basic-Programme nicht erreichbar; er kann nur Maschinenprogramme speichern. Anstelle der RAMs lassen sich auch EPROMs oder sogar ein PIO-Baustein adressieren – auch gemischt geht's. Die Batteriepufferung ist selbstverständlich nur für RAMs notwendig.

Bei beiden Schaltungen kann es zu Störungen kommen, wenn die Busse der Z-80-CPU von weiteren Zusatzschaltungen belastet werden. Dann helfen Treiberbausteine weiter: ein 74LS245 für den Datenbus, zwei 74LS241 für den Adreßbus und drei UND-Gatter eines 74LS08 für die Signale $\overline{RD}$, $\overline{WR}$ sowie $\overline{MREQ}$.

Wird die Schaltung nach Bild 2 hinter einer Speichererweiterung aufgesteckt, arbeitet sie nur dann einwandfrei, wenn bei der RAM-Erweiterung alle benutzten Leitungen durchgeschleift sind. Auf keinen Fall darf die Schaltung gemeinsam mit Peripheriegeräten betrieben werden, die selbst den Adreßbereich zwischen 8 KByte und 16 KByte vollständig nutzen (Doppeladressierung).

Das zeigt die Praxis

Die Schaltung nach Bild 1 eignet sich gut zum Einbau in den ZX 81. Sie erzwingt freilich immer einen Betrieb gemeinsam mit dem 16-KByte-RAM-Modul, da der ZX 81 ohne dieses Modul keinen Speicher mehr zum Ablegen der Systemvariablen vorfindet. Das Ergebnis wäre ein Balkengewirr am Bildschirm. Für einen Test kann man z. B. POKE 33000, 99 eingeben. PRINT PEEK 33000 muß dann den Wert 99 auf den Bildschirm bringen – auch nach NEW oder einem Drücken der Reset-Taste!

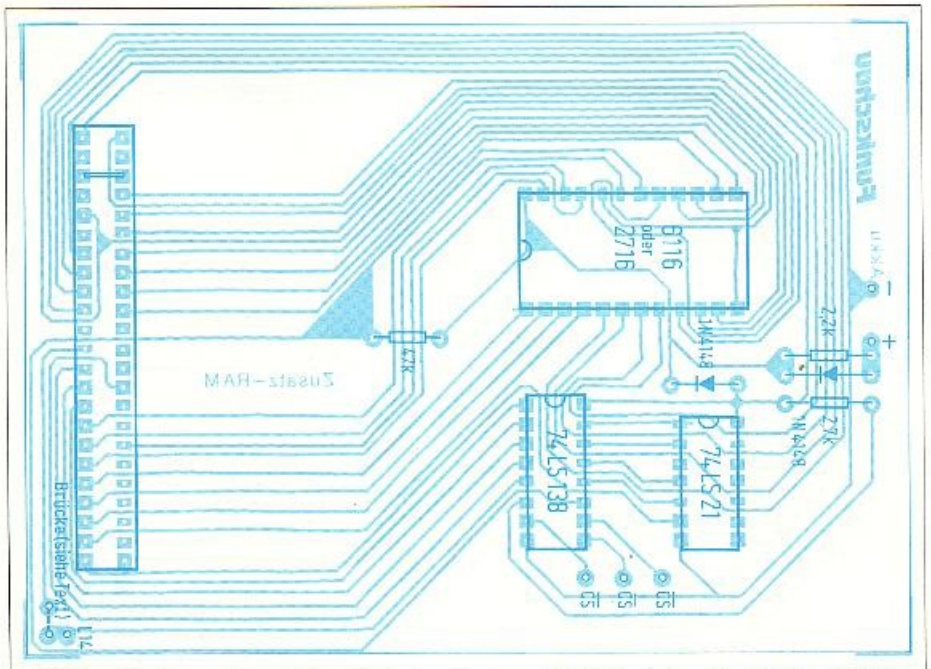
Leider sind Maschinenprogramme oberhalb der Adresse 32 767 nicht lauffähig. Wurde ein Programm dorthin gerettet, dann muß man es nach einem Systemabsturz in einen Speicherbereich unterhalb dieser Adresse kopieren. Erst dann läßt sich das Programm erneut starten.

Beide Probleme kennt die Schaltung nach Bild 2 nicht! Sie hat dafür ein anderes. Da die Schaltung nicht zum Einbau vorgesehen ist, wird man sie zwangsläufig an der Schnittstelle des ZX 81 anschließen. Geschieht das unmittelbar, ist alles in Ordnung. Steckt jedoch ein RAM-Modul (Memotech) dazwischen, fehlt plötzlich das Signal A14. Dieses Signal muß man sich dann mit einer eigenen Leitung aus dem Computer holen. Die Brücke auf der Platine darf in diesem Fall nicht eingelötet werden, denn das Signal A14 darf nicht auf die *Steckerseite* des RAM-Moduls gelangen (Systemabsturz), sondern allein über die ZX-81-Schnittstelle auf die Buchsenseite des Moduls.

Während die Schaltung nach Bild 1 Zusatzspeicher im Adreßbereich 32 768 bis 33 792 schafft, liegt das RAM der Schaltung nach Bild 2 im Adreßbereich 8192 bis 10 240. Ein Test läßt sich mit einer dieser Adressen wie zuvor beschrieben durchführen.

Wer mit 2 KByte RAM nicht auskommt, kann bis zu vier 6116 ICs parallel schalten, d. h., Pin für Pin übereinander löten (stapeln). Lediglich die $\overline{CS}$-Anschlüsse sind dann getrennt mit den Ausgängen des Adreßdecoders auf der Platine zu verbinden. Werden für die RAM-ICs CMOS-Ausführungen gewählt, ist sogar eine Batterie- bzw. Akku-Pufferung möglich. Die Platine ist dafür vorbereitet, Anschlußhinweise geben der Schalt- und Bestückungsplan. R_v ist so zu bemessen, daß der Ladestrom etwa 1/50 der Akkukapazität beträgt (Ladungserhaltung). Im Akkubetrieb legt der 47-kΩ-Widerstand $\overline{CS}$ des 6116 auf H-Pegel um das RAM abzuschalten (Standby). Anderenfalls wäre der Akku bald leer.

M. Longobardi, H.-G. Schmitz, S. Schall



④ **Bestückungsplan:** Beim Betrieb mit einem RAM-Modul muß das Signal A14 über eine Extra-Leitung aus dem Computer auf das Lötauge L14 gegeben werden; die Drahtbrücke muß dann fehlen (siehe Text)

ZX-81-Software:

Entschlüsseln von Morsezeichen

Wer Morsesignale nicht selbst entschlüsseln kann, bekommt jetzt mit dem ZX 81 einen Helfer, der empfangene Morsezeichen im Klartext am Bildschirm darstellt.

Mit dem ZX 81 ist es möglich, ohne großen Hardware-Aufwand Morsezeichen zu decodieren. Das erforderliche Programm verlangt eine 16-KByte-RAM-Erweiterung und ist in Maschinensprache geschrieben, denn Basic ist für diese Aufgabe zu langsam. Die Maschinenroutine bietet einen bemerkenswerten Lekkerbissen: Sie paßt sich der Morsegeschwindigkeit an und enthält einen Testmodus, um die Hardware (PLL-Tondecoder) zu prüfen und korrekt einzustellen.

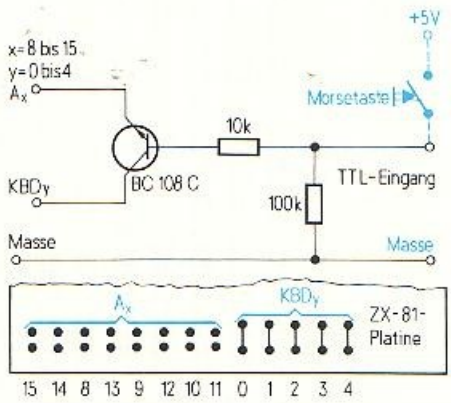
Der Eingabe-Port macht sich auch anderweitig nützlich

Zunächst ist es erforderlich, den ZX 81 mit einem Eingang zu versehen, über den der Computer die Morsezeichen (im TTL-Pegel) empfängt. Der Kassettenrecorder-Eingang EAR ist leider wegen seiner Wechselspannungskopplung hierfür ungeeignet.

Auf sehr einfache Weise läßt sich jedoch die Tastatur als Eingabe-Port nutzen, indem man einen Transistor parallel zu den Anschlüssen einer Taste legt. So kann man sogar zu einer ganzen Reihe von Eingängen kommen, die selbstverständlich auch für andere Zwecke verwendbar sind. Von Basic-Programmen aus lassen sich die Eingänge dann einfach mit der INKEY$-Funktion abfragen; es ist aber nicht zu unterscheiden, ob eine bestimmte Taste gedrückt oder der zugehörige Eingang angesteuert wird.

Bild 1 zeigt die Schaltung für einen Eingang. Die KBD- und A-Anschlüsse (KBD: Keyboard, A: Adresse) sind auf der Lötseite der Computer-Platine schnell zu finden. Welche KBD- und A-Leitungen kombiniert werden, ist für den Morsedecoder gleichgültig, nur die Kombination KBD 0 mit A15 darf nicht gewählt werden (BREAK). Die Leitungskombination, die jeder Taste zugeordnet ist, zeigt *Bild* 2.

Nun muß man nur noch die Morsesignale, die zunächst ja als Ton-Nichtton-Folge vom Funk- oder Radiogerät geliefert werden, in TTL-Signale umwandeln. Falls nicht bereits ein geeigneter Konverter vorhanden ist, kann hierzu ein einfacher PLL-Decoder verwendet werden (*Bild* 3).



① **Eingabe-Port:** An der Tastatur angeschlossen (z. B. an A8 und KBD4) meldet der Port TTL-Signale an den ZX 81 weiter

A_x / KBD_y	8	9	10	11	12	13	14	15
0	SHIFT	A	Q	1	0	P	NEW LINE	SPACE BREAK
1	Z	S	W	2	9	O	L	•
2	X	D	E	3	8	I	K	M
3	C	F	R	4	7	U	J	N
4	V	G	T	5	6	Y	H	B

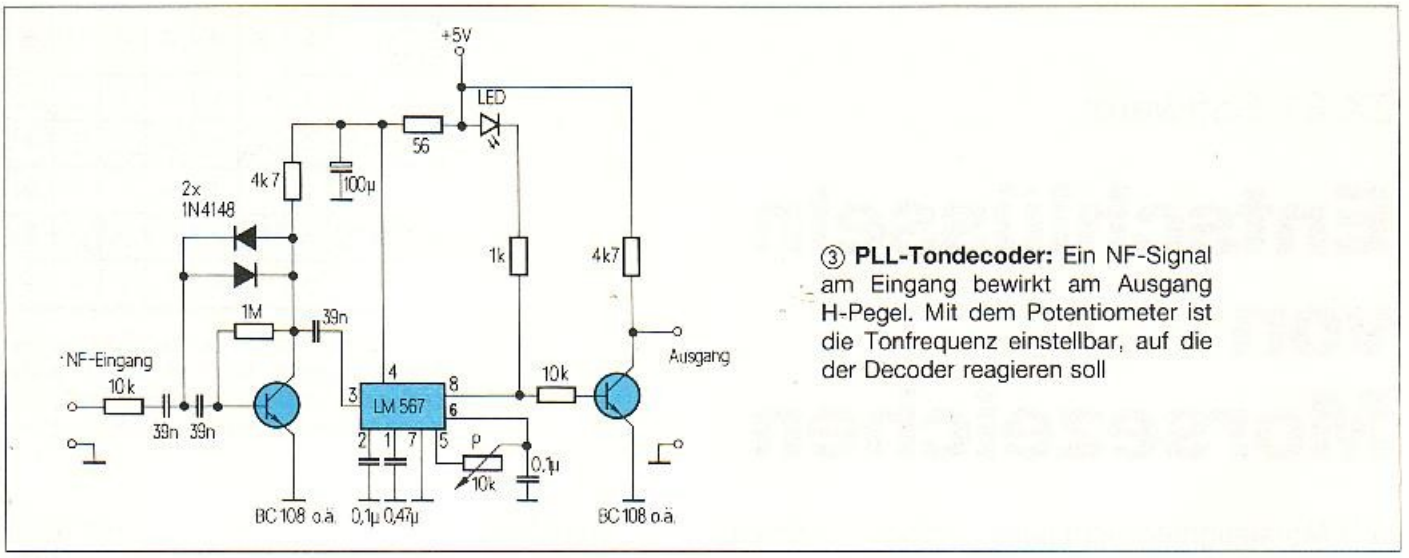
② **Tastaturmatrix:** Ist der Port an den Matrixleitungen A8 und KBD4 angeschlossen, läßt er sich vom Basic aus durch Abfragen (INKEY$) der Taste V ansprechen

Da die Schaltung sehr empfindlich ist, sollte man zum Unterdrücken von Störimpulsen unbedingt selektive Filter vorschalten. Werden andere Decoder verwendet, so ist darauf zu achten, daß der Ausgang auf H-Pegel liegen muß, wenn ein Tonsignal am Eingang anliegt.

Maschinenroutinen lösen alle Aufgaben

In *Bild* 4 ist oben das Basic-Listing des Programms wiedergegeben. Hinter dem REM-Befehl in Zeile 6 müssen mindestens 448 beliebige Buchstaben oder Ziffern eingetastet werden, um Speicherplatz für den Maschinencode zu reservieren. Nach Eintippen des Basic-Programms exakt nach Vorlage muß der Hex-Code mit einem Eingabeprogramm (Seite 72) in diese REM-Zeile gebracht werden (Startadresse: 16 613).

Der Hex-Code enthält eine 50 Byte lange Tabelle, die Auskunft über den Aufbau jedes Morsezeichens gibt, und die Maschinensprache-Routinen, die die Bildschirm-Ausgabe, die Tastaturabfrage und das eigentliche Decodieren übernehmen. Für Leser, die an Einzelheiten interessiert sind, haben wir ein kommentiertes Assembler-Listing vorbereitet, das gegen 4 DM in Briefmarken vom Franzis-Software-Service erhältlich ist. Vor einem ersten Test sollte das Programm auf Band gesichert werden, denn es besteht die Gefahr eines Systemabsturzes, falls ein Fehler bei der Eingabe des Maschinencodes gemacht wurde.



③ **PLL-Tondecoder:** Ein NF-Signal am Eingang bewirkt am Ausgang H-Pegel. Mit dem Potentiometer ist die Tonfrequenz einstellbar, auf die der Decoder reagieren soll

Das Programm ist „lernfähig"

Durch RUN wird das Programm gestartet. Zu Prüfzwecken kann man nun über die Tastatur Zeichen eingeben, indem man eine beliebige Taste (außer SPACE, diese Taste bricht das Programm ab) als Morsetaste benutzt. Mit der Original-Folientastatur ist eine gezielte Eingabe freilich kaum möglich, besser wäre eine am Input-Port angeschlossene Morsetaste.

Die ersten eingegebenen Punkt- und Strichzeichen werden nicht decodiert und angezeigt, sondern ausgewertet, um die Morsegeschwindigkeit zu bestimmen. Erst nach etwa 15 s tauchen die ersten decodierten Zeichen am Bildschirm auf. Falls nicht, wurde bei der Hexcode-Eingabe ein Fehler gemacht, oder das Basic-Programm nicht genau nach Vorlage eingetippt.

Wenn es soweit klappt, wird das Programm gestoppt und neu gestartet. Nun gibt man einen Morse-Text vom Radio, Funkgerät oder Kassettenrecorder über die Zusatz-Hardware auf den Input-Port.

Während des Decodierens darf keine Taste gedrückt werden! Wenn nicht nach wenigen Sekunden die ersten decodierten Zeichen auf dem Bildschirm erscheinen oder wenn Unsinniges ausgegeben wird (z. B. nur die Buchstaben E und T), erhält der Computer wahrscheinlich ein unsauberes Signal.

Um das näher zu untersuchen, unterbricht man wieder den Programmablauf und startet das Programm neu, allerdings diesmal mit RUN 8. Jetzt meldet der ZX 81 mit Punktlinien am Bildschirm fortwährend, welcher Pegel momentan am Eingabe-Port anliegt: Eine magere Punktfolge steht für L-Pegel, eine fette Punktfolge für H-Pegel. Die Folge der fetten Punkte, die zu einem Morse-Punkt- oder -Strich gehören, darf nicht durch magere Punkte unterbrochen sein! Ist das dennoch der Fall, arbeitet der Ton-Decoder nicht einwandfrei.

Selbstverständlich muß zum korrekten Entschlüsseln auch das Format der Morsezeichen stimmen: Ein Strich muß zwei- bis viermal so lange wie ein Punkt dauern, die Pause zwischen den Punkten und Strichen eines Zeichens darf höchstens zwei Punktlängen ausmachen, und die Pause zwischen zwei Buchstaben muß zwischen zwei und vier Punktlängen dauern. Längere Pausen bewirken zusätzlich die Ausgabe eines Leerzeichens.

Schließlich sollte noch die Morsegeschwindigkeit halbwegs konstant sein; allmählichen Tempo-Änderungen paßt sich das Programm indes an.

Michael Schramm

```
1 REM MORSEDECODER
2 REM (C) FUNKSCHAU
3 REM MICHAEL SCHRAMM
4 REM START MIT RUN
5 REM TESTEN MIT RUN 8
6 REM 448 ZEICHEN EINTIPPEN
7 RUN USR 16759
8 RUN USR 17041

ADRESSE   HEX-CODE
DEZIMAL
-----------------------------
16613     47 73 29 52 35 2A 2E 37
16621     5E 00 2D 5C 4C 5A 20 30
16629     38 3C 3E 3F 2F 27 23 21
16637     06 17 15 0B 03 10 09 1F
16645     07 18 0A 1B 04 05 08 19
16653     12 0D 0F 02 0E 1E 0C 16
16661     14 13 D9 F5 11 64 00 1B
16669     7A 63 20 FB 18 02 D9 F5
16677     3E 7F DB FE 1F 30 0B AF
16685     D6 FE D9 3C E6 7F E3 7C
16693     E1 C9 CF 0C D9 F5 A7 20
16701     0F 78 D6 1B 38 0A 3E 76
16709     ED 61 28 06 00 F1 D9 C9
16717     7E A7 28 1E 23 7E A7 28
16725     19 ED 5B 0C 40 21 21 00
16733     19 C5 01 F8 02 ED B0 D5
16741     E1 E5 13 70 0E 1F ED B0
16749     E1 C1 04 F1 77 23 CB A8
16757     D9 C9 FD 36 05 80 CD 2A
16765     0A CD 23 41 20 FB CD 7E
16773     42 20 F6 D6 D5 11 00 FF
16781     CD 23 41 28 FB AF 3C 28
16789     E8 CD 7E 42 20 F8 BA 30
16797     08 4F 07 07 B8 79 38 01
16805     51 B8 38 09 5F 7A 07 07
16813     B6 30 02 16 FF 10 D9 7A
16821     17 38 C6 93 30 C3 21 3C
16829     40 06 10 72 23 10 FC 7A
16837     FD 70 21 2A 0C 40 D9 17
16845     57 AF 3C BA 30 07 CD 7E
16853     42 20 F6 18 F5 21 3C 40
16861     06 10 11 00 00 7E 83 5F
16869     8A 93 57 23 10 F7 06 04
16877     4B CB 3A CB 1B 10 F9 16
16885     01 AF 3C B9 30 07 CD 7E
16893     42 28 F7 18 09 AF CD 39
16901     41 CD 23 41 28 FB AF 3C
16909     28 31 CD 7E 42 20 F8 B9
16917     30 19 43 CB 38 B8 38 17
16925     21 3C 40 C5 06 00 FD 4E
16933     21 09 0C CB A1 FD 71 21
16941     77 C1 37 CB 12 38 A6 AF
16949     3C B9 30 07 CD 7E 42 28
16957     F7 18 CB 7A FE 1A 20 04
16965     3E 26 18 0E FE 11 20 04
16973     3E 34 18 05 FE 1C 20 09
16981     3E 3A CD 39 41 3E 2A 18
16989     1A FE 10 20 09 3E 28 CD
16997     39 41 3E 20 18 0D 21 E5
17005     40 01 32 00 ED B1 20 06
17013     3E 3F 91 CD 39 41 C3 DA
17021     41 F5 AF CD 17 41 28 01
17029     3C CD 17 41 28 01 3C A7
17037     E3 7C E1 C9 06 00 2A 0C
17045     40 D9 CD 7E 42 3E 1B 28
17053     02 3E 04 CD 39 41 18 F2
```

④ **Software:** Zuerst ist das Basic-Programm einzutippen. Anschließend wird mit einem Eingabeprogramm der Hexcode im REM-Kommentar von Zeile 6 untergebracht

ZX 81 intern:

Das ROM hilft gerne mit

Im ROM des ZX 81 sind zahlreiche fix und fertige Routinen, die nicht nur vom Betriebssystem, sondern auch von eigenen Maschinenprogrammen aufgerufen werden dürfen. So läßt sich z. B. eine Liste der in einem Basic-Programm verwendeten Zeilennummern im Nu zusammenstellen.

Dieser Beitrag wendet sich an fortgeschrittenere ZX-81-Programmierer, denen das Wort Maschinensprache keine Gänsehaut über den Rücken jagt. Wer hier nicht sattelfest ist, das Programm aber dennoch nutzen will, der kann es mit dem Hexcode-Eingabeprogramm (Seite 72) eingeben. Wer sich überdies mit der Maschinensprache des ZX 81 vertraut machten möchte, um die einzelnen Programmteile zu verstehen, dem sei das FUNKSCHAU-Sonderheft „Klartext für den ZX 81" ans Herz gelegt, das der vermeintlich schwierigen Maschinensprache ihre Schrecken nimmt.

Schritt für Schritt zur Problemlösung

Mittlerweile gibt es für den ZX 81 kommentierte „ROM-Listings". Das sind Bücher, die dem Kenner der Z-80-Maschinensprache Einblick in das Betriebssystem und den Basic-Interpreter des ZX 81 geben. Man stößt dort auf zahlreiche Routinen, die sich gut für eigene Zwecke nutzen lassen und mit *call* oder *rst* (Restart) aufzurufen sind.

```
ADR. MNEMONIC          HEX-CODES
4082 LD HL,00000       210000
4085 PUSH HL           E5
4086 LD HL,16509       217D40
4089 CALL 02546        CDF209
408C LD (16507),DE     ED537B40
4090 CALL 02725        CDA50A
4093 LD A,027          3E1B
4095 RST 10            D7
4096 RST 10            D7
4097 RST 10            D7
4098 RST 10            D7
4099 POP HL            E1
409A INC HL            23
409B PUSH HL           E5
409C XOR A             AF
409D LD A,(IY+057)     FD7E39
40A0 CALL 16568 *      CDB840
40A3 LD HL,(16507)     2A7B40
40A6 LD A,(HL)         7E
40A7 CP 040            FE28
40A9 JR C,$40B9        38DE
40AB XOR A             AF
40AC CALL 16568 *      CDB840
40AF LD B,021          0615
40B1 LD C,018          0E12
40B3 CALL 02293        CDF508
40B6 POP BC            C1
40B7 RET               C9
40B8 ADD A,(IY+058)    FDB63A
40BB CP 005            FE05
40BD JR NC,$40CA       300B
40BF CALL 00699        CDBB02
40C2 LD A,L            7D
40C3 CP 254            FEFE
40C5 JR NZ,$40BF       20FB
40C7 CALL 03086        CD0E0C
40CA RET               C9
```

Listing: Dieses Programm nutzt ROM-Routinen und zeigt sämtliche in einem Basic-Programm vergebenen Zeilennummern am Bildschirm in Kolonnen an.

Wir wollen jetzt einige dieser Routinen nutzen, um ein Maschinenprogramm zu entwickeln, das sämtliche in einem Basic-Programm verwendeten Zeilennummern blitzschnell am Bildschirm auflistet. Das verkürzt die Suche nach freien Zeilennummern, wenn man noch eine Programmergänzung unterbringen möchte. Berauschend groß ist dieser Vorteil zwar gerade nicht; dafür läßt sich an diesem Anwendungsbeispiel gut zeigen, wie sich durch Aufruf der einen oder anderen Routine die Aufgabe immer besser lösen läßt.

Für den Anfang genügt schon ein nur 22 Byte langes Maschinenprogramm, um sich die Basic-Zeilennummern auflisten zu lassen. Es hat lediglich eine feste Startadresse und kann daher über RAMTOP oder in einer REM-Zeile „gefahren" werden, ohne verändert werden zu müssen, da nur mit relativen Sprüngen gearbeitet wird. Die Befehle lauten:

Hexcodes	Mnemonics
21 7D 40	*ld hl,16509*
CD F2 09	*call 2546*
ED 53 7B 40	*ld (16507),de*
CD A5 0A	*call 2725*
2A 7B 40	*ld hl,(16507)*
7E	*ld a,(hl)*
FE 28	*cp 40*
D0	*ret nc*
18 ED	*jr-19*

Wenn im Registerpaar hl die erste Adresse der ersten Basic-Zeile steht, ermittelt *call 2546* die Startadresse der nächsten Zeile und stellt sie im Registerpaar DE zur Verfügung. Nach der Ablage in 16507 wird sie mit *call 2725* auf den Bildschirm gebracht. Gleichzeitig wird die Printposition auf den ersten freien Platz nach dem Ausdruck verschoben.

Nach dem Aufruf mit RAND USR... erhält man dann sämtliche Zeilennummern, die für ein Basic-Programm verwendet worden sind. Solange es sich um maximal dreistellige Zahlen handelt, ist das noch lesbar. Bei vierstelligen Zahlen entsteht aber ein wahrer Zahlenwust, weil keine Lücken gelassen werden. Darum eine erste Verbesserung, die Übersicht schafft.

Durch *rst 16* wird ein im Akku a stehendes Zeichen auf den Bildschirm gebracht. Also fügen wir nach *call 2725* ein:

3E 1B	*ld a,27*
D7	*rst 16*
D7	*rst 16*
D7	*rst 16*
D7	*rst 16*

Der relative Sprung am Schluß muß jetzt jr −25 lauten (18 E7). Und schon erscheint auf dem Bildschirm anstelle der zweiten, vierten, sechsten und achten Kolonne eine Reihe von vier Punkten. Wer will, kann durch *xor a* vier Leerstellen setzen oder irgend etwas anderes in den Akku laden.

Was aber, wenn die Programme so lang sind, daß der Bildschirm nicht alles faßt? Das ist der Fall ab 96 Zeilen. Dann muß durch eine weitere Ergänzung dafür gesorgt werden, daß die Liste „gescrollt" wird. Dazu folgende Ergänzung, die nach den vier rst-Befehlen einzufügen wäre (jetzt nicht mehr mitprogrammieren, sondern am Schluß gleich das komplette Listing eintippen):

AF	*xor a*
FD 86 39	*add a,(iy+57)*
FD 86 3A	*add a,(iy+58)*
FE 05	*cp 5*
38 0A	*jr c,+10*

und bei der Sprungadresse:

CD BB 02	*call 699*
7D	*ld a,l*
FE FE	*cp 254*
20 F8	*jr nz,−8*
CD 0E 0C	*call 3086*
18 E9	*jr −23*

Die zweite Befehlsfolge führt mit den vier ersten Befehlen eine Tastaturabfrage durch, die durch Drücken einer Taste der Tastengruppe Z, X, C, V den Scroll auslöst und wieder in das laufende Programm zurückspringt. Das Listing der Zeilennummern wird dann so lange nach oben geschoben, wie eine der genannten Tasten festgehalten wird oder das Programm abgelaufen ist.

Wer bis hier mitgearbeitet hat, wird merken, daß die Sprungadressen nicht mehr passen. Das liegt daran, daß in das Programm auch noch eine Zählschleife eingebaut wurde; denn wer möchte nicht gern wissen, wieviele Zeilen das Basic-Programm nun insgesamt hat? Dazu kommt vorweg die Folge:

21 00 00	*ld hl,0*
E5	*push hl*

und dann die eingangs genannte Befehlsfolge *ld hl,16509* usw. bis zum vierten rst-Befehl. Danach wird noch eingefügt:

E1	*pop hl*
23	*inc hl*
E5	*push hl*

Anstelle von *ret nc* ist jetzt ein Sprungbefehl nötig. Zugleich wird die Scrollroutine universell ausgelegt, so daß mit dem Aufruf der Basic-Zeile:

PRINT USR 16514;"ZEILEN"

in Zeile 21 des Bildschirms die Zahl der verwendeten Basic-Zeilen direkt ablesbar ist, sobald das Maschinenprogramm abgelaufen ist.

Das *Bild* zeigt das komplette Listing für den Einbau in eine REM-Zeile (72 Byte) ab Adresse 16514. Wenn das Programm an einer anderen Stelle arbeiten soll, dann sind die beiden markierten Werte anzupassen. Martin Rötter

ZX-81-Hardwaretip:

Lange Leitung

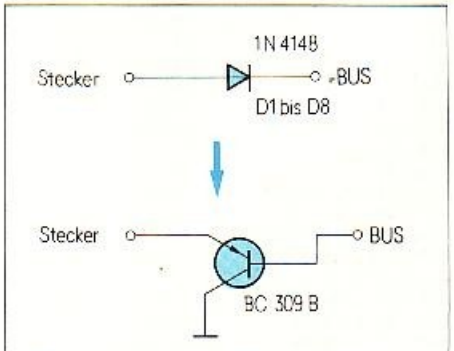
Wer seinen ZX 81 mit einer externen Tastatur betreibt wird gewiß auch schon die Erfahrung gemacht haben, daß man die Tastatur höchstens 50 cm vom Computer absetzen darf. Schuld daran ist das Verbindungskabel zwischen beiden Einheiten, das bei größerer Leitungslänge den Adreßbus des ZX 81 zu stark (kapazitiv) belastet. Wird der ZX 81 mit Peripheriegeräten betrieben, die den Adreßbus ohnehin belasten, dann kann es sogar schon bei Leitungslängen unter 50 cm zu Störungen kommen.

Die Belastung des Adreßbusses durch die Tastatur läßt sich sehr einfach verringern, indem die Tastaturdioden D1 bis D8 auf der Platine des ZX 81 durch die Basis-Emitter-Strecken von PNP-Transistoren (z. B. BC 309 B) ersetzt werden. Legt man die Kollektoranschlüsse der Transistoren auf Masse, so haben wir es mit Transistoren in Kollektor-Schaltung zu tun, deren Arbeitswiderstände im Netzwerk RP 3 versteckt sind (*Bild*).

Der Adreßbus wird jetzt um den Stromverstärkungsfaktor der Transistoren geringer belastet, so daß Leitungslängen von etwa 1 m zulässig sind. Da es nun aber möglich ist, die Pull-up-Widerstände in RP 3 zu verringern (z. B. durch Parallelschalten eines 10-k-Arrays) sind sogar noch erheblich längere Leitungen erlaubt. So ergaben Versuche mit einem geschirmten 5 m langen Tonband-Verlängerungskabel (starke kapazitive Last!) einwandfreie Ergebnisse.

Den ZX 81 über ein langes Kabel außer Reichweite zu bringen ist insbesondere bei Exemplaren nützlich, die auf Erschütterungen mit dem „Aussteigen" reagieren. Ursache dieser Störungen ist die Kontaktleiste an der Rückseite des ZX 81. Hier kommt es bei angeschlossenen Peripheriegeräten (z. B. Speichererweiterungen) leider oft zu Kontaktschwierigkeiten, auf die der ZX 81 mit Befehlsverweigerung reagiert. Eine Behelfslösung ist das Reinigen der Kontakte mit einem „harten" Radiergummi, der die isolierende Oxidschicht auf den Kontaktflächen entfernen kann.

Hans-Peter Recktenwald



Schaltungskniff: Werden die Tastaturdioden durch Transistoren ersetzt, lassen sich externe Tastaturen über längere Kabel anschließen

ZX-81-Softwaretip:

Stopper für LIST

Sind sehr lange Maschinenprogramme in einer REM-Zeile untergebracht, dann kann es nach dem Kommando LIST zu einem „Endloslisten" dieser Programmzeile kommen, das sich nicht mehr stoppen läßt. Dieses Problem wird beseitigt, wenn vor dem eigentlichen Maschinenprogramm in den Speicherzellen 16 514/16 515 die Hex-Codes 76 und C9 (dezimal: 118/201) untergebracht werden.

Dann ist das gesamte Programm nach dem REM-Befehl am Bildschirm verschwunden, obwohl es nach wie vor im Speicher steht und mit einer um zwei Byte verschobenen Startadresse (z. B. 16 516) aufzurufen ist. Durch den NEW-LINE-Code (76h) sieht das Betriebssystem des ZX 81 aber keinen Grund mehr, das Programm, das danach kommt, zu listen. Paul Webranitz

Arbeitshilfe:

Raumgestalter für Maschinencode

REM-Zeile blitzschnell erzeugt

Maschinenprogramme, in einer REM-Zeile abgelegt, haben den Vorteil, daß man sie auf Band speichern und wieder laden kann. Dazu muß man eine REM-Zeile mit genügend Platz für das Maschinenprogramm erstellen. Bei längeren Programmen ist das eine zeitraubende, fehlerträchtige und vor allem langweilige Angelegenheit.

Es liegt nun nahe, ein Programm zu schreiben, das einem eine beliebig lange REM-Zeile für Maschinenprogramme erzeugt. Mancherorts werden solche Programme auch schon angeboten. Leider handelt es sich dabei meist um sehr umfangreiche Basic-Programme, die zudem noch sehr langsam sind.

10 KByte in 14 Sekunden

Hier soll hingegen ein Programm vorgestellt werden, das zum einen im Programmspeicher insgesamt nur 105 Byte Platz beansprucht, also kurz ist, und zum anderen eine REM-Zeile für 10 KByte innerhalb von 14 s reserviert. Das gesamte Programm besteht aus zwei Teilen: einem zehn Zeilen langen Basic-Programm, und einem 43 Byte umfassenden Maschinenprogramm, das in der ersten REM-Zeile des Basic-Programms verborgen ist.

Um das Maschinenprogramm in die REM-Zeile zu bekommen, benutzen wir einen „Dezimal-Loader" (*Bild 1*). Wir tippen das Programm ein und achten zum allerletzten Mal darauf, daß in der REM-Zeile genau 43 Zeichen stehen. Anschließend starten wir das Programm mit RUN und geben die Dezimalwerte des Maschinenprogramms (*Bild 2*) ein.

```
   1 REM 00000000000000000000000
00000000000000000000
   2 FOR N=16514 TO 16556
   3 PRINT N;"   ";
   4 INPUT X
   5 POKE N,X
   6 PRINT X,
   7 NEXT N
```

① **Dezimal-Lader:** Hilfsprogramm zur Eingabe von Maschinenprogrammen in Dezimalwerten

Nach Eingabe des letzten Dezimalwertes drücken wir NEWLINE und stellen fest, daß in der Zeile 1 das Maschinenprogramm steht. Dann werden noch die Zeilen 2 bis 10 durch die entsprechenden Zeilen in *Bild 3* ersetzt bzw. ergänzt. Dabei müssen wir uns ganz genau an die Vorlage halten und dürfen keinesfalls bis einschließlich Zeile 6 die Länge des Programms verändern. Soll das „Kind" in der zweiten Zeile einen anderen Namen erhalten, so muß dieser aus genau drei Zeichen bestehen.

Falls ein Speichermodul an unserem ZX 81 angeschlossen ist, läßt sich durch Eingabe von POKE 16389,68 der SAVE- und LOAD-Vorgang um 16 Sekunden verkürzen.

Unbedingt notwendig ist es jedoch, den Rechner vor dem SAVE-Vorgang in den FAST-Modus zu schalten, da sonst bei Ausführung des Maschinenprogramms ein Systemabsturz sehr wahrscheinlich ist. Bei der Funktionsbeschreibung des Maschinenprogramms wird der Grund noch erklärt. Um das Programm nun auf Kassette zu speichern, startet man den Recorder und läßt das Programm mit RUN ablaufen. Anschließend schalten wir den Computer aus und wieder ein und laden das Programm von der Kassette.

Der Rechner befindet sich danach im FAST-Modus und wartet auf die Eingabe der Anzahl der zu reservierenden Bytes. Nach der Eingabe wird der Bildschirm kurze Zeit dunkel, worauf die Meldung 0/7 erscheint. Die gewünschte REM-Zeile ist damit erstellt.

16514	237	16515	91
16516	50	16517	64
16518	42	16519	219
16520	64	16521	25
16522	34	16523	219
16524	64	16525	33
16526	221	16527	64
16528	35	16529	213
16530	62	16531	28
16532	205	16533	38
16534	5	16535	225
16536	1	16537	1
16538	0	16539	167
16540	237	16541	66
16542	235	16543	32
16544	239	16545	33
16546	88	16547	0
16548	34	16549	127
16550	64	16551	33
16552	218	16553	64
16554	54	16555	0
16556	201		

② **Maschinenprogramm:** Den Speicherplätzen 16514 bis 16556 werden Dezimalwerte zugeordnet

Vorsicht, Falle!

Falls mehr als 695 Bytes reserviert werden soll, treten automatisch die Programmzeilen 8 und 9 in Aktion. Die beiden POKE-Befehle verhindern, daß der Computer in eine endlose LIST-Schleife geschickt wird, die nur durch Ziehen des Netzsteckers wieder verlassen werden kann.

Wenn wir nun „1" und NEWLINE drücken und damit die Programmzeile 1 löschen, erscheint auf dem Bildschirm die gewünschte REM-Zeile mit der Zeilennummer 0. Danach löschen wir ebenfalls noch die Programmzeilen 7 bis 10. Wurde eine REM-Zeile mit mehr als 695 Zeichen reserviert, sieht man auf dem Bildschirm nur die Zeile 0 mit dem Befehl REM. Trotzdem enthält diese Zeile die geforderte Anzahl von Zeichen. In diesem Fall funktioniert auch das Kursor-Down-Kommando nicht; um die eventuell nachfolgenden Programmzeilen zu sehen, muß LIST 1 eingegeben werden.

Nun kann ab Zeilennummer 1 ein beliebiges Programm eingegeben werden. Um dabei nicht ständig nur die Zeile 0 zu sehen, gibt man ein: LIST N und POKE 16419, N. Dabei ist N die Nummer der ersten Zeile.

Programm mit Raffinesse

Um die Funktionsweise des vorgestellten Programms kennenzulernen, schauen wir uns *Bild 3* an. In Zeile 1 gelangt unser Maschinenprogramm (von Bild 1) und Zeile 2 versteht sich von selbst. Nach dem Laden startet das Programm mit der Ausführung der 3. Zeile und fragt nach der Anzahl der zu reservierenden Bytes.

Nun folgt ein kleiner Kunstgriff: Die Variable N wird im Variablen-Bereich als 5-Byte-Fließkommazahl abgespeichert. Da es sich jedoch um eine positive ganze Zahl handelt, die zudem sicher kleiner als 65536 (= 2^{16}) ist, wäre eine 2-Byte-Darstellung viel zweckmäßiger. Eine 2-Byte-Zahl könnte man nämlich ohne Schwierigkeiten in ein Doppelregister des Mikroprozessors laden. Die Funktion RAND benutzt die Systemvariable SEED (siehe Sinclair-Handbuch), die aus zwei Bytes besteht. Will man

```
   1 REM : REM-ZEILEN-GENERATOR
0000000000000000000000
   2 SAVE "REM"
   3 INPUT N
   4 RAND N
   5 RAND USR 16514
   6 REM
   7 IF N<695 THEN GOTO 10
   8 POKE 16606,118
   9 POKE 16607,118
  10 SLOW
```

③ **REM-Zeilen-Generator:** Das komplette Programm zur Erzeugung von beliebig langen REM-Zeilen

dem Zufallszahlengenerator einen definierten Anfangswert X zuteilen, so gibt man den Befehl RAND X. Dieser Befehl bewirkt aber nichts weiter, als daß der Systemvariablen SEED der Wert X in 2-Byte-Form zugeteilt wird.

In Zeile 4 unseres Programms erhält SEED auf diese Weise die Anzahl der zu reservierenden Bytes. Zeile 5 ruft das Maschinenprogramm auf. Hier darf unbesorgt ebenfalls der Befehl RAND verwendet werden, da zunächst das Maschinenprogramm vollständig abgearbeitet wird und erst danach der Inhalt des bc-Registers des Mikroprozessors der Systemvariablen SEED zugeteilt wird.

Die 6. Zeile ist die entscheidende REM-Zeile, in der der gewünschte Speicherplatz vom Maschinenprogramm reserviert wird. Zeile 7 bewirkt, wenn die Zahl der zu reservierenden Bytes kleiner als 695 ist, den Sprung zur Zeile 10. Bei mehr als 694 zu reservierenden Bytes wird durch die Programmzeilen 8 und 9,

```
4082 LD DE,(16434)        ED5B3240
4086 LD HL,(16603)        2ADB40
4089 ADD HL,DE            19
408A LD (16603),HL        22DB40
408D LD HL,16605          21DD40
4090 INC HL               23
4091 PUSH DE              D5
4092 LD A,028             3E1C
4094 CALL 01318           CD2605
4097 POP HL               E1
4098 LD BC,00001          010100
409B AND A                A7
409C SBC HL,BC            ED42
409E EX DE,HL             EB
409F JR NZ,$4090          20EF
40A1 LD HL,00088          215800
40A4 LD (16511),HL        227F40
40A7 LD HL,16602          21DA40
40AA LD (HL),000          3600
40AC RET                  C9
```

④ **Assembler-Listing:** Da eine ROM-Routine genutzt wird, ist das Programm nur 43 Byte lang

wie erwähnt, die endlose LIST-Schleife vermieden.

Was macht nun aber das Maschinenprogramm (*Bild 4*)? Wie wir der Gliederung des Speichers auf Seite 171 des Sinclair-Handbuchs entnehmen können, liegt das Programm gleich hinter dem Systemvariablen-Bereich. Wird der Programmbereich durch das Verlängern der REM-Zeile vergrößert, müssen die Inhalte des Bildspeichers, des Variablenbereiches, des Arbeitsraumes und des Kalkulatorstapels nach oben verschoben werden. Anschließend müssen die Systemvariablen D-FILE, VARS, E-LINE, STKBOT und STKEND entsprechend neue Werte erhalten.

Glücklicherweise muß dies nicht alles programmiert werden, da das Betriebssystem im ROM eine Routine enthält, die diese Arbeit erledigt. Diese Routine verschiebt den gesamten Speicherinhalt von der Adresse, auf die das Registerpaar hl zeigt, bis zu der Stelle, die von der Systemvariablen STKEND angezeigt wird, um ein Zeichen nach oben. Dabei werden alle Systemvariablen, deren Werte größer als das Registerpaar hl sind, um 1 erhöht. Die freigewordene Speicherzelle erhält anschließend den Wert des a-Registers.

Unser Maschinenprogramm ruft nun diese Routine so oft auf, bis Platz für die gewünschte Anzahl von Bytes geschaffen ist. Vorher gibt es den beiden Bytes, die die Länge der REM-Zeile angeben, den neuen Wert. Außerdem erhält die 1. Programmzeile als Länge einen Wert, der der Gesamtlänge der Programmzeilen 1 bis 5 entspricht. Dadurch werden später beim Löschen der Zeile 1 auch die Zeilen 2 bis 5 gelöscht. Darüber hinaus bekommt die erzeugte REM-Zeile die Zeilennummer 0, um die Zeile vor versehentlichem Löschen zu schützen.

Michael Naujoks

Programmieren in Maschinensprache:

Z-80-Training mit dem ZX 81

Kontrolle der Registerinhalte

Programmieren in Maschinensprache will gelernt sein; insbesondere das Wissen um die Veränderung von Registerinhalten durch Programme ist wichtig. Das Programm „Z-80-Trainer" gibt Hilfestellung bei den ersten Schritten.

Das Programm „Z-80-Trainer" für den ZX 81 mit 16 KByte RAM ist eine visuelle Lernhilfe für den Anfänger und Einsteiger in die Z-80-Maschinensprache. Der Anwender kann bis zu acht Maschinenbefehle in hexadezimaler Form in den Computer eingeben, worauf mit Hilfe des Trainers alle wesentlichen Registerinhalte in hexadezimaler und dualer Form angezeigt werden.

Die duale Darstellung der einzelnen Bits des Flag-Registers ist dabei besonders interessant, da man sofort sehen kann, welche Bits (Carry-Bit, Zero-Bit, Sign-Bit usw.) gesetzt sind und welche nicht. Davon hängt ja bekanntlich bei den bedingten Sprungbefehlen der weitere Verlauf eines Programmes ab.

Insofern ist der „Z-80-Trainer" auch zum begrenzten Austesten kleinerer Routinen geeignet, wenn man lediglich sehen will, ob auch die Register so verändert werden wie man es sich gedacht hat.

Was kann der Trainer?

Folgende Arten von Maschinenbefehlen lassen sich verarbeiten:

- 8-Bit- und 16-Bit-Ladebefehle
- arithmetische/logische Befehle mit 8 oder 16 Bit Länge
- Rotations- und Schiebebefehle
- Einzelbitbefehle (set, res, bit)

Dabei werden vom „Trainer" insgesamt die Inhalte von zehn Registern auf dem Bildschirm dargestellt: a, f, b, c, d, e, h, l, ix und iy.

```
16514   ED 4B C1 41 C5 F1 ED 4B
16522   C3 41 ED 5B C5 41 2A C7
16530   41 DD 2A C9 41 FD 2A CB
16538   41 CD B8 41 ED 43 C3 41
16546   ED 53 C5 41 22 C7 41 F5
16554   C1 ED 43 C1 41 DD 22 C9
16562   41 FD 22 CB 41 2A 0C 40
16570   11 B3 00 19 EB 2A C1 41
16578   CD 8B 41 2A C3 41 CD BB
16586   41 2A C5 41 CD BB 41 2A
16594   C7 41 CD 94 41 21 5D 00
16602   19 EB 2A C9 41 E5 7C CD
16610   A4 41 13 E1 7D CD A4 41
16618   21 3F 00 19 EB 2A CB 41
16626   E5 7C CD A4 41 13 E1 7D
16634   CD A4 41 2A 0C 40 11 AA
16642   00 19 3A C2 41 CD 71 41
16650   11 09 00 19 3A C1 41 CD
16658   71 41 11 29 00 19 3A C4
16666   41 CD 71 41 11 09 00 19
16674   3A C3 41 CD 71 41 11 29
16682   00 19 3A C6 41 CD 71 41
16690   11 09 00 19 3A C5 41 CD
16698   71 41 11 29 00 19 3A CB
16706   41 CD 71 41 11 09 00 19
16714   3A C7 41 CD 71 41 11 4D
16722   00 19 3A CA 41 CD 71 41
16730   3A C9 41 CD 71 41 11 32
16738   00 19 3A CC 41 CD 71 41
16746   3A CB 41 CD 71 41 C9 06
16754   08 CB 27 D4 B5 41 DC 7F
16762   41 23 10 F5 C9 F5 3E 1D
16770   77 F1 C9 F5 3E 1C 77 F1
16778   C9 CD 94 41 21 30 00 19
16786   EB C9 E5 7C CD A4 41 21
16794   10 00 19 EB E1 7D CD A4
16802   41 C9 F5 CB 3F CB 3F CB
16810   3F CB 3F C6 1C 12 13 F1
16818   E6 0F C6 1C 12 C9 00 00
16826   00 00 00 00 00 00 C9 00
```

① **Hex Listing:** Für den Maschinenspracheteil des „Z-80-Trainers"

```
   1 REM 331 BELIEBIGE ZEICHEN
  25 FAST
  30 RAND USR 16514
 100 PRINT AT 0,12;"Z-80 TRAINER
"
 105 PRINT AT 1,12;"------------
",AT 5,0;"A -"
 110 PRINT ,,"B -"
 120 PRINT ,,"D -"
 130 PRINT ,,"H -"
 140 PRINT AT 5,17;"F -",AT 7,17
;"C -",AT 9,17;"E -"
 145 PRINT AT 11,17;"L -",AT 14,
2;"IX -",AT 16,2;"IY -"
 150 PRINT AT 21,0;"EINGABE:"
 160 PRINT AT 21,9;
 170 LET F=1
 180 LET Z=1
 190 LET USER=16824
 200 FOR X=16824 TO 16831
 210 POKE X,0
 220 NEXT X
 225 SLOW
 230 IF INKEY$<>"" THEN GOTO 230
 240 IF INKEY$="" THEN GOTO 240
 250 LET A$=INKEY$
 260 IF A$="." THEN GOTO 430
 270 IF CODE A$<>118 THEN GOTO 3
00
 280 IF F=1 THEN GOTO 410
 290 GOTO 230
 300 IF CODE A$>43 OR CODE A$<28
THEN GOTO 230
 310 PRINT A$;
 320 IF F=1 THEN LET S=CODE A$
 330 LET F=-F
 340 IF F=-1 THEN GOTO 230
 350 LET T=CODE A$
 360 POKE USER,((S-28)*16+T-28)
 370 LET USER=USER+1
 380 LET Z=Z+1
 390 IF Z=9 THEN GOTO 410
 400 GOTO 230
 410 FAST
 420 RAND USR 16514
 430 PRINT AT 21,9;"
   "
 450 GOTO 160
 500 SAVE "Z80"
 510 RUN
```

② **Basic Listing:** In Zeile 1 gehört das Maschinenprogramm

Vom Vertauschen der 16-Bit-Register (exx) und der beiden af-Registerpaare (exaf,af') wurde abgesehen, ebenso von der Darstellung des i- und des r-Registers, da sie für den „Z-80-Trainer" nicht von Bedeutung sind und die Anschaulichkeit des Programmes darunter nicht leidet.

Der Maschinenspracheteil des Programmes läuft im FAST-Modus, um Manipulationen des ix- und des iy-Registers zu ermöglichen.

Das Programm besteht aus einem Maschinensprache- und einem Basic-Teil. Zunächst werden in die Programmzeile 1 mindestens 331 beliebige Zeichen eingegeben (s. Seite 55). Dann werden die Bytes des Hex-Listings aus *Bild 1* mittels eines Hex-Ladeprogramms (s. Seite 72) in die REM-Zeile geladen.

Anschließend tippt man das Basic-Programm (*Bild 2*) ein. Das fertige Programm wird anschließend mit GOTO 500 auf Band geladen. Danach startet das Programm automatisch, BREAK hält es jederzeit wieder an.

Kontrolle durch Prüfsumme

Um die korrekte Eingabe des Maschinenprogramms zu kontrollieren, kann man die Prüfsumme des Hex-Listings bilden; sie beträgt 34275. Mit dem Programm in *Bild 3* läßt sich die Prüfsumme ermitteln. Die Ermittlung der Prüfsumme sollte erfolgen, noch bevor der „Z-80-Trainer" das erste Mal gestartet wurde bzw. wenn alle Register auf dem Bildschirm leer sind, da die Speicherstellen ab Adresse 16824 durch das laufende Programm verändert werden.

Nach dem Laden vom Band startet das Programm automatisch. Alle Register sind leer, die Eingabezeile steht zur Aufnahme von maximal 8 Byte Maschinensprache zur Verfügung. Die Befehle werden hexadezimal eingegeben und die Eingabe mit NEWLINE beendet. Daraufhin erscheinen umgehend auf dem Bildschirm die neuen Registerinhalte.

Vorsicht ist bei *push*- und *pop*-Befehlen geboten: Wenn mit *push* ein Byte oder mehrere Bytes auf den Stapel gelegt werden, müssen diese auch vor dem Ende der Routine mit *pop* wieder vom Stapel geholt werden, ansonsten bricht das Programm zusammen.

Fehlerhafte Eingaben lassen sich durch Drücken der Punkttaste (.) zurücknehmen. Werden genau 8 Byte eingegeben, startet die Programmausführung automatisch.

```
1000 FAST
1010 LET S=0
1020 FOR X=16514 TO 16633
1030 LET S=S+PEEK X
1040 NEXT X
1050 SLOW
1060 PRINT "PRUEFSUMME = ";S
```

③ **Prüfsummen-Programm:** Dient zur Kontrolle der Hex-Eingabe in die REM-Zeile

Probe aufs Exempel

Alle Register sind leer. Der Akkumulator (a-Register) soll mit 100 (64h) und das b-Register mit 151 (97h) geladen, anschließend b von a subtrahiert werden. Nachfolgend ist der Inhalt des Doppel-Registers af nach ix zu kopieren.

```
            Z-80 TRAINER
            ------------

A - 11001101 CD   F - 10011111 9F
B - 10010111 97   C - 00000000 00
D - 00000000 00   E - 00000000 00
H - 00000000 00   L - 00000000 00

  IX - 1100110110011111   CD9F
  IY - 0000000000000000   0000

EINGABE:
```

④ **Beispiel:** Bildschirm nach Abarbeitung des Beispiels

Im Z-80-Assembler lauten die einzelnen Programmschritte folgendermaßen:

3E 64	*ld a,64h*
06 97	*ld b,97h*
90	*sub a,b*
F5	*push af*
DDE1	*pop ix*

Ein return-Befehl (C9) wird vom Programm selbständig erzeugt.

Die einzelnen Bytes werden der Reihe nach in die Eingabezeile eingegeben: 3E64069790F5DDE1. Die Programmausführung beginnt automatisch, weil insgesamt acht Bytes eingegeben werden. Daraufhin werden die Registerinhalte gemäß *Bild 4* angezeigt. Die Subtraktion ist erfolgt. Die Differenz von 100 und 151 (CDh) -51 steht in hexadezimaler Form im Akku. Das b-Register enthält weiterhin den ursprünglich geladenen Wert. Der Inhalt des Doppel-Registers af wurde nach ix übertragen, natürlich bleibt er auch in af erhalten. Im Flag-Register sind u. a. Sign-Flag als Anzeige für ein negatives Ergebnis und Subtract-Flag als Anzeige für eine Subtraktion gesetzt. Das Programm hat funktioniert!

Heinz-Werner Siedentopf

Alte Batterien nutzen:

Wieder in Dienst stellen

Wenn einen morgens nicht mehr der Radiowecker mit Getöse aus dem Bett wirft, ist es an der Zeit, ihm neue Batterien zu spendieren. Vom alten, jetzt überflüssigen Batteriesatz muß man sich indes nicht sofort trennen, denn er stellt noch genügend Energie bereit, um Geräte mit sehr kleiner Leistungsaufnahme zu betreiben. Solche Geräte sind z. B. Digitaluhren oder Taschenrechner mit LC-Anzeige.

Einziges Problem: die Batterien müssen in das Gerät passen. Deshalb sollte man z. B. beim Kauf von (billigen) Taschenrechnern darauf achten, daß diese nicht mit teuren Knopfzellen betrieben werden, sondern mit gängigen Batterien.

Die Probe aufs Exempel wurde an einem Taschenrechner mit zwei „alten" Mignonzellen gemacht: Zwar nur selten benutzt, rechnet er auch nach Monaten noch brav, nur der Kontrast der Anzeige hat nachgelassen. Und ein technischer K. o. durch ausgelaufene Batterien ist unwahrscheinlich. -ll

ZX-81-Softwaretip:

Gegen hektischen Bildaufbau

Bei langen Programmen in Basic kommt es unweigerlich immer wieder zu der unangenehmen Situation, daß die Zeile, die man gerade neu eingibt, nicht mehr auf den Bildschirm paßt. Der ZX 81 baut das Bild dann in mehreren Anläufen auf, was durchaus schon einige Sekunden dauern kann und recht nervenaufreibend ist. Auch der Befehl LIST xx funktioniert im allgemeinen nicht, weil der Computer schon beim nächsten NEWLINE wieder macht, was er will.

Ein einfaches und wirksames Gegenmittel ist die Anweisung POKE 16420,100. Sie bewirkt, daß die Zeile, in der der Cursor vor der Anweisung stand, jetzt die erste angezeigte Zeile ist – und man hat seine Ruhe. H. Kling

Arbeitshilfe:

Schieber für Maschinencode

Von einem Speicherbereich in den anderen

Wie verschiebe ich ein Maschinenprogramm aus einer REM-Zeile in den Speicherbereich oberhalb von RAMTOP? Wie ändere ich dabei kritische Adressen? All das nimmt einem dieses Verschiebe-Programm ab.

Oftmals gibt es die Notwendigkeit, ein in Maschinensprache geschriebenes Programm in einen anderen Speicherbereich zu verschieben. Beim ZX 81 kann das beispielsweise deshalb notwendig sein, weil man selbst eine andere Speichererweiterung besitzt als derjenige, der das Programm geschrieben hat, und das Programm in einem möglichst hohen Adreßbereich im RAM liegen soll. Leider ist es mit dem bloßen Verschieben des Maschinensprache-Programms nicht getan; im allgemeinen müssen auch etliche Adressen im Maschinencode verändert werden. Dies ist nötig bei direkten Sprüngen, Unterprogrammaufrufen sowie direkten und indirekten Ladebefehlen.

Automatische Verschiebung mit Komfort

Das hier beschriebene Verschiebe-Programm nimmt einem die ganze Arbeit des Verschiebens und Anpassens von Z-80-Maschinenprogrammen ab. Das Basic-Listing des Programms ist in *Bild 1* wiedergegeben. Hinter dem REM in Zeile 1 sind 390 beliebige Buchstaben oder Ziffern zur Reservierung von Speicherplatz für den Maschinensprache-Teil des Programms einzutasten (s. S. 55). Anschließend wird mit einem Hex-Lader (s. S. 72) der Maschinencode (*Bild 2*) eingegeben. Dabei ist darauf zu achten, daß die Startadresse zur Eingabe hier den Wert 16591 hat.

```
   1 REM 390 BELIEBIGE ZEICHEN
  10 PRINT "VERSCHIEBEPROGRAMM FU
ER","Z80-MASCHINENCODE"
  20 PRINT
  30 LET A=16591+64
  40 PRINT "LAENGE DES PROGRAMMS
: ";
  50 GOSUB 300
  60 PRINT "STEHT IM SPEICHER AB
ADR.: ";
  70 GOSUB 300
  80 PRINT "LAUFFAEHIG AB ADR.: "
;
  90 GOSUB 300
 100 LET L=X
 110 PRINT "NEUE LAUFADR.: ";
 120 INPUT X
 130 PRINT X
 140 LET X=X-L
 150 IF X<0 THEN LET X=X+65536
 160 GOSUB 320
 170 PRINT "SUCHEN AB ADR.: ";
 180 GOSUB 300
 190 PRINT "SUCHEN BIS ADR (EXKL.
): ";
 200 GOSUB 300
 210 PRINT "VERSCHIEBUNG DURCHFUE
HREN (J/N)?"
 220 INPUT X$
 230 POKE A+175,24
 240 IF X$="J" THEN POKE A+175,6
 250 PRINT "PROGRAMM LAEUFT."
 260 LET X=USR A
 270 PRINT "ENDE."
 280 PRINT X;" ADRESSEN VERAENDER
T."
 290 IF X$="J" THEN PRINT "VERSCH
IEBUNG DURCHGEFUEHRT."
 295 STOP
 300 INPUT X
 310 PRINT X
 320 LET M=INT (X/256)
 330 POKE A,X-256*M
 340 POKE A+1,M
 350 LET A=A+2
 360 RETURN
```

① **Basic-Listing:** In der ersten Programmzeile wird das Maschinenprogramm untergebracht

Das Verschiebe-Programm und das Nutzprogramm müssen sich natürlich gleichzeitig im Speicher befinden. Beim ZX 81 geht man so vor, daß man zunächst das Nutzprogramm lädt und in einen Speicherbereich kopiert, in dem es durch das anschließende Laden des Verschiebe-Programms nicht gelöscht wird. Nachdem das Verschiebe-Programm durch RUN gestartet worden ist, erwartet das Programm einige Angaben über die zu erledigende Aufgabe:

① Länge des Nutzprogramms in Bytes

② Anfangsadresse im Speicher

③ Adresse, ab der es stehen muß, um zu funktionieren

④ neue Adresse, ab der es lauffähig sein soll

⑤ Adresse, ab der verändert werden soll

⑥ Adresse, bei der die Bearbeitung enden soll

⑦ Angabe darüber, ob das veränderte Programm an die neue Laufadresse kopiert werden soll

Die Eingaben unter ⑤ und ⑥ ermöglichen es, nur in Teilbereichen des Programms Änderungen vorzunehmen, um zum Beispiel Tabellen nicht zu verändern. Kennt man das Programm nicht näher, so gibt man unter ⑤ den Wert aus ② ein und unter ⑥ die Summe aus ① und ②.

Das Verschiebe-Programm arbeitet einige Sekunden und liefert als Ergebnis

```
16591   75 59 55 59 76 59 56 59
16599   7E 59 5E 59 7E 48 5E 59
16607   55 51 55 51 55 51 55 51
16615   55 51 55 51 00 04 55 51
16623   55 51 55 51 55 51 55 51
16631   55 51 55 51 55 51 55 51
16639   DD 5B 9D 7B 9D 5B 9D 5B
16647   5D 5B 5D 5B 5D 5B 5D 5B
16655   66 17 9A 68 9A 68 B6 E5
16663   9A 68 00 B0 01 00 00 D9
16671   E5 2A 17 41 AF 47 4F 57
16679   7E FE ED 20 0F E5 23 7E
16687   1E 04 E6 C7 FE 43 28 31
16695   1E 02 18 2D FE DD 28 04
16703   FE FD 20 03 0C 23 7E E5
16711   CB 3F CB 10 1F CB 10 21
16719   CF 40 5F 19 7E CB 20 28
16727   04 CB 3F 10 FC E6 03 5F
16735   20 01 1C 0D 20 03 1F 30
16743   F9 E1 E5 19 CB 53 D9 E1
16751   23 20 22 28 7E FE C3 28
16759   1C FE CD 28 18 E6 C7 FE
16767   C2 28 12 FE C4 28 0E 7E
16775   E6 CF FE 01 28 07 7E E6
16783   E7 FE 22 20 26 23 5E 23
16791   56 E5 2A 13 41 E5 37 ED
16799   52 E1 30 16 C5 ED 4B 0F
16807   41 09 C1 ED 52 38 0B 2A
16815   15 41 19 EB E1 72 2B 73
16823   03 E5 E1 D9 EB 2A 19 41
16831   7A B3 28 07 37 ED 52 EB
16839   D2 23 41 06 22 2A 13 41
16847   ED 4B 15 41 09 ED 4B 0F
16855   41 ED 5B 11 41 E5 ED 52
16863   E1 30 05 EB ED B0 18 07
16871   09 EB 09 2B 1B ED B8 E1
16879   D9 C9
```

② **Maschinencode:** Diese Hex-Zahlen werden mit Hilfe eines Hex-Laders in die REM-Zeile übertragen

der USR-Funktion die Anzahl der veränderten Adressen. Das Nutzprogramm steht auf jeden Fall noch im selben Speicherbereich wie vorher, jedoch mit veränderten Adressen im Maschinencode, und wurde außerdem, falls mit ⑦ gewünscht, an die neue Laufadresse verschoben.

Das Nutzprogramm in Maschinensprache wird durch das Verschiebe-Programm Befehlszeile für Befehlszeile untersucht. Das Verschiebe-Programm läßt sich in zwei Teile aufgliedern: Der erste Teil stellt fest, wie lang der gerade anstehende Befehl mit einem bestimmten Operationscode ist (eins, zwei, drei oder vier Bytes). Hierbei stützt sich das Programm auf eine Liste, die für jeden Op-Code zwei Bits an Informationen enthält. Der zweite Teil untersucht, ob es sich um einen Befehl mit 2-Byte-Adreßteil handelt (durch Vergleich mit bestimmten Masken). Bei positivem Ergebnis wird geprüft, ob die Adresse innerhalb des Programms liegt. Falls ja, wird die Adresse um den Verschiebe-Betrag geändert.

```
411B LD BC,00000        010000
411E EXX                D9
411F PUSH HL            E5
4120 LD HL,(16663)      2A1741
4123 XOR A              AF
4124 LD B,A             47
4125 LD C,A             4F
4126 LD D,A             57
4127 LD A,(HL)          7E
4128 CP 237             FEED
412A JR NZ,$413B        200F
412C PUSH HL            E5
412D INC HL             23
412E LD A,(HL)          7E
412F LD E,004           1E04
4131 AND 199            E6C7
4133 CP 067             FE43
4135 JR Z,$4168         2B31
4137 LD E,002           1E02
4139 JR $4168           1B2D
413B CP 221             FEDD
413D JR Z,$4143         2804
413F CP 253             FEFD
4141 JR NZ,$4146        2003
4143 INC C              0C
4144 INC HL             23
4145 LD A,(HL)          7E
4146 PUSH HL            E5
4147 SRL A              CB3F
4149 RL B               CB10
414B RRA                1F
414C RL B               CB10
414E LD HL,16591        21CF40
4151 LD E,A             5F
4152 ADD HL,DE          19
4153 LD A,(HL)          7E
4154 SLA B              CB20
4156 JR Z,$415C         2804
4158 SRL A              CB3F
415A DJNZ $4158         10FC
415C AND 003            E603
415E LD E,A             5F
415F JR NZ,$4162        2001
4161 INC E              1C
4162 DEC C              0D
4163 JR NZ,$416B        2003
4165 RRA                1F
4166 JR NC,$4161        30F9
416B POP HL             E1
4169 PUSH HL            E5
416A ADD HL,DE          19
416B BIT 2,E            CB53
416D EXX                D9
416E POP HL             E1
416F INC HL             23
4170 JR NZ,$4194        2022
4172 DEC HL             2B
4173 LD A,(HL)          7E
4174 CP 195             FEC3
4176 JR Z,$4194         281C
417B CP 205             FECD
417A JR Z,$4194         281B
417C AND 199            E6C7
417E CP 194             FEC2
4180 JR Z,$4194         2812
4182 CP 196             FEC4
4184 JR Z,$4194         280E
4186 LD A,(HL)          7E
4187 AND 207            E6CF
4189 CP 001             FE01
418B JR Z,$4194         2807
418D LD A,(HL)          7E
418E AND 231            E6E7
4190 CP 034             FE22
4192 JR NZ,$41BA        2026
4194 INC HL             23
4195 LD E,(HL)          5E
4196 INC HL             23
4197 LD D,(HL)          56
419B PUSH HL            E5
4199 LD HL,(16659)      2A1341
419C PUSH HL            E5
419D SCF                37
419E SBC HL,DE          ED52
41A0 POP HL             E1
41A1 JR NC,$41B9        3016
41A3 PUSH BC            C5
41A4 LD BC,(16655)      ED4B0F41
41A8 ADD HL,BC          09
41A9 POP BC             C1
41AA SBC HL,DE          ED52
41AC JR C,$41B9         38DB
41AE LD HL,(16661)      2A1541
41B1 ADD HL,DE          19
41B2 EX DE,HL           EB
41B3 POP HL             E1
41B4 LD (HL),D          72
41B5 DEC HL             2B
41B6 LD (HL),E          73
41B7 INC BC             03
41B8 PUSH HL            E5
41B9 POP HL             E1
41BA EXX                D9
41BB EX DE,HL           EB
41BC LD HL,(16665)      2A1941
41BF LD A,D             7A
41C0 OR E               B3
41C1 JR Z,$41CA         28D7
41C3 SCF                37
41C4 SBC HL,DE          ED52
41C6 EX DE,HL           EB
41C7 JP NC,16675        D22341
41CA JR $41EE           1822
41CC LD HL,(16659)      2A1341
41CF LD BC,(16661)      ED4B1541
41D3 ADD HL,BC          09
41D4 LD BC,(16655)      ED4B0F41
41D8 LD DE,(16657)      ED5B1141
41DC PUSH HL            E5
41DD SBC HL,DE          ED52
41DF POP HL             E1
41E0 JR NC,$41E7        3005
41E2 EX DE,HL           EB
41E3 LDIR               EDB0
41E5 JR $41EE           18D7
41E7 ADD HL,BC          09
41E8 EX DE,HL           EB
41E9 ADD HL,BC          09
41EA DEC HL             2B
41EB DEC DE             1B
41EC LDDR               EDB8
41EE POP HL             E1
41EF EXX                D9
41F0 RET                C9
```

③ **Assembler-Listing:** Mit diesem Maschinenprogramm werden Programme verschoben und Adressen verändert

Grenzen dieses Verfahrens

Fast immer arbeitet das Programm zur Zufriedenheit. Unter bestimmten Bedingungen muß es jedoch zwangsläufig versagen, im wesentlichen dann,

- falls Tabellen mit absoluten Adressen vorhanden sind, die geändert werden müssen, oder
- falls 2-Byte-Zahlen, die nicht geändert werden dürfen, zufällig im Adreß-Bereich des Programms liegen.

Treten Schwierigkeiten auf, so bleibt einem nichts anderes übrig, als den Bereich, der den Fehler verursacht, mit einem Disassembler näher zu untersuchen, den Verschieber gezielt nur auf Programm-Teilbereiche anzusetzen und gegebenenfalls Tabellen oder einzelne Adressen von Hand zu ändern.

Auch für andere Z-80-Computer

Das Programm kann auf jedem Z-80-Computer mit Basic-Betriebssystem eingesetzt werden. Das Maschinenprogramm muß in einen anderen geeigneten Speicherbereich verlegt werden (eigentlich eine Aufgabe für das Programm selbst), was mittels eines Assemblers ja leicht möglich ist. Vor dem *ret* am Ende des Programms (*Bild 3*) muß der Inhalt von bc in ein anderes Registerpaar geladen werden, falls ein anderes als das bc-Registerpaar das Ergebnis der USR-Funktion liefert.

Im Basic-Listing ist nur die Zeile 30 zu ändern. Der Variablen A wird die Anfangsadresse des Maschinencodes plus 64 zugewiesen. Michael Schramm

Messen mit dem ZX 81:

NF-Frequenzmesser

Mit einem kurzen Programm und einer minimalen Hardware-Ergänzung wird der ZX 81 zum NF-Frequenzmesser. Außer der direkten Anzeige verspricht die Weiterverarbeitung des Ergebnisses viele nützliche Anwendungen.

Der Frequenzbereich, in dem der ZX 81 als Frequenzmesser eingesetzt werden kann, reicht von etwa 1 Hz bis 20 kHz. Frequenzwert und Taktverhältnis des Meßsignals lassen sich dann unmittelbar am Bildschirm ablesen. Außer der Anwendung als Meßgerät für Frequenzen eröffnen sich für Hobby-Elektroniker jedoch noch zahlreiche weitere interessante Perspektiven.

Hier einige Anregungen: Stammt das Meßsignal von einem Oszillator mit einem Widerstand oder Kondensator als frequenzbestimmendes Glied, so ist die gemessene Frequenz ein Maß für den Widerstands- oder Kapazitätswert.
Nach einer Umrechnung durch eine Programmergänzung lassen sich daher auch Widerstände und Kondensatoren messen. Ähnliches gilt für die Temperaturmessung, Feuchtemessung usw. Denkbar wäre außerdem eine Selektivruf-Auswertung per Programm oder eine Meßanordnung, die die Windgeschwindigkeit oder Drehzahlen erfaßt.

So wird gemessen

Bild 1 zeigt das Basic-Listing des Programms. In Zeile 1 sind hinter der REM-Anweisung wenigstens 83 beliebige Buchstaben oder Ziffern einzutasten. Mittels eines Hex-Code-Eingabeprogramms (siehe Seite 72) wird dann der Hex-Code von *Bild 2* in dem durch Zeile 1 reservierten Speicherplatz untergebracht.

Das Programm wird durch RUN gestartet. Nachdem das Wort RECHTECK auf dem Bildschirm erschienen ist, muß innerhalb der nächsten vier Sekunden das zu untersuchende Signal dem ZX 81 zur Verfügung gestellt werden. Dies geschieht am einfachsten über einen Input-Port, wie er beim Aufsatz über den Morsecoder auf Seite 51 beschrieben ist. Auch hier gilt, daß jede Leitungskombination außer KBD0 mit A15 gewählt werden darf.

```
   1 REM HIER MIND. 83 BELIEBIGE
BUCHSTABEN EINTIPPEN
  20 SLOW
  30 REM FREQUENZBEREICH 1-20000
 HERTZ
  40 PRINT "RECHTECK",,USR VAL "
16530"
  50 LET F=VAL "2"
  60 DIM T(F)
  70 FOR X=VAL "16514" TO VAL "1
6528" STEP F
  80 LET F=VAL "3-F"
  90 LET T(F)=VAL "T(F)+PEEK X+2
56*PEEK (X+1)"
 100 NEXT X
 110 PRINT "FREQUENZ: ";VAL "339
000/(T(1)+T(2))";" HERTZ";TAB NO
T PI;"TAKTVERHAELTNIS HIGH/LOW:"
,VAL "T(2)/T(1)"
```

① **Basic-Listing:** Der REM-Kommentar reserviert 83 Speicherzellen für das Maschinenprogramm in Bild 2. Wegen speicherplatzsparender VAL-Funktionen paßt das Programm noch in den 1-KByte-Grundspeicher

Während der Messung sollte keine Taste berührt werden, da sonst das Eingangssignal verfälscht würde. Falls kein Signal vorhanden ist, kann man die Messung durch mehrmaliges Drücken der SPACE-Taste abbrechen. Sobald der Computer aber anfängt, die Meßergebnisse anzuzeigen, darf die Verbindung zwischen Input-Port und Signalquelle wieder gelöst werden.

Sinusförmige Signale lassen sich nur dann messen, wenn sie zuvor in Rechtecksignale umgewandelt werden. Eine einfache Schaltung, die Sinussignale mit einer Spannung von mindestens einigen Millivolt in Rechtecksignale mit nahezu symmetrischem Taktverhältnis umwandelt, zeigt *Bild 3*.

Das „Meßfenster" ist nur vier Perioden breit

Die eigentliche Frequenzmessung läuft in Maschinensprache ab, da Basic viel zu langsam für diese Aufgabe ist. Die Maschinencode-Routine mit Startadresse 16530 wird mittels der USR-Funktion in Zeile 40 aufgerufen.

Zunächst durchläuft die Routine eine Verzögerungsschleife, die etwa vier Sekunden beansprucht. Während dieser Zeit soll, wie schon angegeben, das zu untersuchende Signal auf den Input-Port gelegt werden. Sobald die vier Sekunden verstrichen sind, wird das FAST-Modus eingeschaltet, um maximale Programmablaufgeschwindigkeit zu erreichen; und das Programm beginnt, die Tastatur und damit auch den Input-Port mit einem *in*-Befehl ständig abzufragen.

Um den Anfangspunkt einer Periode zu erwischen, wartet das Programm, bis die nächste negative Flanke des Eingangssignals auftritt. Daraufhin wird über Verzögerungsschleifen die Dauer der nächsten acht Halbperioden gemessen.

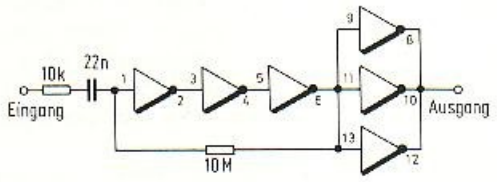
Die Ergebnisse werden jeweils als Zwei-Byte-Zahl in den 16 Byte langen Speicherbereich geschrieben, der direkt hinter dem REM-Schlüsselwort von Zeile 1 mit Adresse 16514 beginnt und unmittelbar vor dem eigentlichen Maschinenprogramm endet. Die Frequenzmessung selbst dauert also (je nach Frequenz) nur einen kleinen Sekundenbruchteil, nämlich die Dauer von vier Perioden des Eingangssignals.

Trotzdem wird eine recht hohe Meßgenauigkeit erreicht. Allgemein gilt, daß die Meßgenauigkeit mit zunehmender Frequenz abnimmt. Liegt die Frequenz

```
ADR. MNEMONIC          HEX-CODES

4092 LD DE,01024       110004     Etwa 4 s
4095 DJNZ $4095        10FE       Wartezeit
4097 DEC DE            1B         durch
4098 LD A,D            7A         Zeitschleife.
4099 OR E              B3
409A JR NZ,$4095       20F9
409C LD D,127          167F       FAST.
409E OUT 253,A         D3FD       Input-Port
40A0 IN A,254          DBFE       abfragen
40A2 INC A             3C         und auf
40A3 AND D             A2         negative Flanke
40A4 JR Z,$40A0        28FA       des Eingangssi-
40A6 LD BC,00254       01FE00     gnals warten.
40A9 IN A,(C)          ED78
40AB INC A             3C
40AC AND D             A2
40AD JR NZ,$40A9       20FA
40AF LD SP,16530       319240     SP auf Ablage-Bereich.
40B2 LD E,004          1E04       E ist Perioden-Zähler.
40B4 LD HL,00000       210000     HL mißt Halbperioden.
40B7 IN A,(C)          ED78       Input-Port abfragen.
40B9 INC A             3C
40BA INC HL            23         Warten bis
40BB AND D             A2         Pegel wechselt.
40BC JR Z,$40B7        28F9
40BE PUSH HL           E5         Halbperiodendauer auf
40BF LD HL,00000       210000     Stack.
40C2 IN A,(C)          ED78       Zweite Halbperiode
40C4 INC A             3C         messen.
40C5 INC HL            23
40C6 AND D             A2
40C7 JR NZ,$40C2       20F9
40C9 PUSH HL           E5         Halbperiodendauer auf Stack.
40CA DEC E             1D         Beide Schleifen viermal durchlaufen.
40CB JR NZ,$40B4       20E7
40CD LD SP,(16386)     ED7B0240   SP auf (ERR-SP).
40D1 OUT 254,A         D3FE       SLOW.
40D3 RST 08            CF
40D4 DFB FF            FF         Error-Routine im ROM aufrufen.
```

② **Assembler-Listing:** Die Hex-Codes (mittlere Spalte) werden ab der Adresse 16530 (4092h) im REM-Kommentar (Bild 1) untergebracht



③ **Sinus-/Rechteck-Wandler:** Die Anschlußnumerierung der Inverter gilt für das IC CD 4069 (Pin 14: +5 V; Pin 7: Masse)

außerhalb des Meßbereichs, so werden falsche Ergebnisse ermittelt.

Nach der Messung wird die Maschinenroutine verlassen und das Basic-Programm weiter ausgeführt. In der FOR-NEXT-Schleife, bestehend aus den Zeilen 70 bis 100, werden dann die Werte der LOW-Halbperioden in T(1) aufaddiert ebenso wie die Werte der HIGH-Perioden in T(2). Anhand beider Summen wird schließlich in den Zeilen 110 und 120 Frequenz und Taktverhältnis des Eingangssignals errechnet. Die Zahl 339 000 ergibt sich aus der Arbeitsgeschwindigkeit des Z-80-Prozessors und damit aus der durch einen Keramikschwinger vorgegebenen Taktfrequenz.

Michael Schramm

Ersatz für PAUSE-Befehl:

Flimmerfrei

Das ZX-81-Basic verfügt über einen PAUSE-Befehl, der sich zudem in seiner Zeitdauer durch einen Zahlenwert variieren läßt. Leider bringt aber die Verwendung des Befehls ein lästiges Flakkern des Bildschirms mit sich. Deshalb wird häufig der PAUSE-Befehl mit einer FOR-NEXT-Schleife umgangen. Aber auch dieser Weg hat einen Nachteil.

Eine wirklich mit PAUSE gleichwertige Funktion enthält noch den Abbruch der Pause durch Drücken einer Taste. Und diese Möglichkeit ist bei älteren ZX 81 auch nicht unkritisch: Im SLOW-Modus kann der Rechner „abstürzen", wenn unmittelbar vor Ende einer Pause eine Taste gedrückt wird.

All die bisher beschriebenen Nachteile hat das kleine Unterprogramm nicht, dessen Listing im *Bild* zu finden ist. Es ist dort in ein Programmbeispiel eingebaut, damit gleich der Nutzen zu erkennen ist.

```
  10 LET PAUSE=1010
  20 LET DAUER=50
  30 PRINT ".";
  40 GOSUB PAUSE
1000 REM PAUSE
1010 FOR N=0 TO DAUER STEP 1.5
1020 IF INKEY$<>"" THEN LET N=DAU
ER
1030 NEXT N
1040 RETURN
```

PAUSE-Ersatz: Mit dem Unterprogramm ab Zeile 1000 kann bei einer Pause das lästige Bildflimmern vermieden werden

Das Unterprogramm PAUSE wird mit der Programmzeile 40 GOSUB PAUSE aufgerufen, wobei die Variable PAUSE in Zeile 10 den entsprechenden Wert zugewiesen bekommt. Der Variablen DAUER wird vorher die gewünschte Pausendauer zugewiesen, wobei der in unserem Beispiel verwendete Wert 50 etwa einer Sekunde entspricht. Sollten dabei geringe zeitliche Differenzen auftreten, so können sie durch Veränderung des STEP-Wertes (Zeile 1010) ausgeglichen werden. Natürlich können den Variablen PAUSE und DAUER auch andere, kürzere Namen zugeordnet werden.

Wolf-Dieter Roth

ZX-81-Softwaretip:

Brückenbauer

Anknüpfen von Basic-Programmen

Der ZX 81 kennt keinen speziellen LOAD-Befehl, der zu einem bereits geladenen Programm ein zweites Programm hinzulädt. Mit zwei Hilfsroutinen läßt sich diese Aufgabe aber dennoch lösen. Dabei rechtfertigt der Nutzen den Aufwand.

Wer selber viel programmiert, wird das Problem kennen: An ein Programm, an dem man gerade arbeitet, ließe sich ein schon früher geschriebenes (Unter-) Programm gut anfügen. Normalerweise bleibt dann nichts anderes übrig, als die Ergänzung Zeile für Zeile geduldig abzutippen. Das fordert Zeit, die viel besser zur Fertigstellung des eigentlichen Hauptprogramms spendiert werden sollte.

Rettungsinsel für Programme

Aber wie nur läßt sich die Ergänzung einfach hinzuladen, wo doch der ZX 81 beim Ausführen des LOAD-Befehls das eigentliche Hauptprogramm erbarmungslos aus dem Programmspeicher herauswirft? Hier hilft nur noch ein Trick weiter.

Das Hinzuladen von Programmen – andere Computer haben dafür spezielle Ladebefehle (z. B. MERGE oder ENTER) – ist beim ZX 81 allein auf Umwegen möglich. Der Systemvariablen RAMTOP fällt dabei eine Schlüsselrolle zu: Bekanntlich sind alle Speicherplätze oberhalb von RAMTOP sicher vor dem Zugriff des ZX-81-Betriebssystems, also auch sicher vor der Zwangslöschung durch Befehle wie NEW oder LOAD. Ein oberhalb von RAMTOP abgelegtes Programm wird daher nicht durch Hinzuladen eines zweiten Programms gelöscht, wie das normalerweise durch die LOAD-Funktion verursacht wird.

Nunmehr wären zwar die beiden Programme im RAM des Computers, aber sie sind noch nicht aneinander angehängt. Um das zu bewerkstelligen, muß man mit den Programmen etwas jonglieren. Und damit es dabei keine Mißverständnisse gibt, geben wir ihnen die Namen P1 und P2:

P1 sei das Hauptprogramm, an das das Programm P2 angefügt werden soll. P1 ist oberhalb von RAMTOP abgelegt, wogegen P2 durch Hinzuladen seinen Platz im üblichen Programmspeicher gefunden hat. Jetzt gilt es, P2 um genauso viele Bytes im Programmspeicher nach hinten (oben) zu schieben, daß sich P1 nahtlos in die entstandene Lücke hineinkopieren läßt. Dann stehen beide Programme in der richtigen Reihenfolge, erst P1 und dann P2, gemeinsam im Programmspeicher – das Ziel ist erreicht. So kompliziert wie es scheint, ist das alles gar nicht, wenn wir zum Jonglieren Hilfsroutinen verwenden.

Reger Verkehr im RAM

Der Brückenschlag zwischen zwei Programmen erfordert zwei Hilfsroutinen, die einmal RAMTOP herabsetzen (*Bild 1*) und dann das Verschieben und Kopieren erledigen (*Bild 2*). Beide Hilfsroutinen sind zunächst abzutippen und, von einer rd. 5 s dauernden Pause getrennt, auf Band aufzuzeichnen (GOTO 190 bzw. GOTO 240).

Den Umgang mit den Routinen probieren wir jetzt am besten gleich an einem Beispiel aus: An das „Hauptprogramm" (P1)

```
10 PRINT "HAUPTPROGRAMM"
```

soll das „Unterprogramm" (P2)

```
20 PRINT "UNTERPROGRAMM"
```

angefügt werden. Tippen wir also das Prograämmchen P1 ein. Um es später oberhalb von RAMTOP unterzubringen, müssen wir den Speicherbedarf dieses Programms kennen. Dazu verhilft die Direktanweisung:

```
PRINT  PEEK  16396
+  256*PEEK 16397–16509
```

die die Differenz zwischen dem Programmspeicherende (D-FILE) und dem Programmspeicherbeginn (Adresse 16509) bildet. Im Beispiel lautet das Ergebnis 21 (Byte).

Jetzt ist P1 auf Kassette abzuspeichern und danach die Hilfsroutine Append 1 (Bild 1) zu laden. Sie fordert zur Eingabe des soeben festgestellten Speicherbedarfs von P1 auf. Geschieht dies, so wird RAMTOP (Normalwert 32768 bei 16 KByte Speichererweiterung) um diesen

```
   1 REM APPEND1
  10 PRINT "KASSETTENRECORDER AU
SSCHALTEN."
  15 PAUSE 200
  20 CLS
  25 PRINT "APPENDPROGRAMM TEIL
1"
  55 PRINT
  60 PRINT "BITTE GEBEN SIE DIE
LAENGE DES  PROGRAMMS EIN (IN BY
TE),DAS","UEBER RAMTOP GESETZT W
ERDEN","SOLL."
  75 PRINT
  80 PRINT "LADEN SIE ANSCHLIESS
END DAS     PROGRAMM APPEND2 VON
 DER","KASSETTE"
 100 INPUT A
 110 LET A=A+50
 120 LET A=32768-A
 130 POKE 16388,A-256*INT (A/256
)
 140 POKE 16389,INT (A/256)
 150 PRINT
 160 PRINT "ACHTUNG: PROGRAMM WI
RD GELOESCHT"
 170 PAUSE 100
 180 NEW
 190 SAVE "A1"
 200 RUN
```

① **Hilfsroutine 1:** Sie reserviert oberhalb von RAMTOP Speicherplatz für das Hauptprogramm und zwei Maschinenprogramme

```
   1 REM APPEND 2
  10 PRINT "APPENDPROGRAMM TEIL
2"
  20 PRINT
  30 PRINT "DIESES PROGRAMM LAED
T 2 MASCHI- NENPROGRAMME UEBER R
AMTOP."
  40 PRINT
  50 PRINT "STARTEN SIE DEN LADE
VORGANG     DURCH TASTENDRUCK."
 100 PAUSE 4E4
 110 PRINT
 115 PRINT "LADER AKTIVIERT"
 120 LET A$="042 012 064 001 125
 064 237 066 068 077 237 067 254
 127 237 091 004 064 033 125 064
 237 176 201 "
 125 LET A$=A$+"033 125 064 237
075 254 127 205 158 009 017 125
064 237 075 254 127 042 004 064
237 176 201 "
 130 FOR A=32719 TO 32765
 140 POKE A,VAL A$( TO 3)
 150 LET A$=A$(5 TO )
 160 NEXT A
 170 CLS
 180 PRINT "LADEVORGANG BEENDET"
 190 PRINT
 195 PRINT
 200 PRINT "- WICHTIG - WICHTIG
- WICHTIG -"
 210 PRINT
 220 PRINT "MASCHINENPROGRAMM 2
(USR 32743) SOLLTE NUR IM"'FAST"
-MODUS AUF- GERUFEN WERDEN"
 230 STOP
 240 SAVE "A■"
 250 RUN
```

② **Hilfsroutine 2:** Die in den Zeilen 120/125 enthaltenen Maschinenprogramme werden oberhalb von RAMTOP untergebracht. Sie bewirken die Verknüpfung der eigentlichen Anwenderprogramme

Wert zuzüglich 50 Byte automatisch tiefergesetzt. Die 50 Byte Zugabe sind nötig, weil die zweite Hilfsroutine Append 2 (Bild 2) Maschinenprogramme enthält, die ebenfalls oberhalb von RAMTOP untergebracht werden (zum Ablegen von P1 über RAMTOP, zum Verschieben von P2 und Kopieren von P1). Das gezielte Tiefersetzen von RAMTOP ist die einzige Aufgabe von Append 1.

Nun erst geht es richtig los, denn jetzt ist das Programm Append 2 zu laden. Es ist im wesentlichen ein Transferprogramm, das die in den Zeilen 120/125 enthaltenen Maschinencodes oberhalb von RAMTOP (Adressen: 32719 bis 32765) unterbringt.

Nach der Meldung „Ladevorgang beendet" darf unser kleines Hauptprogramm P1 wieder von Kassette geladen werden. Mit der Direkteingabe RAND USR 32719 wird es von dem ersten Maschinenprogramm in den reservierten Speicher oberhalb von RAMTOP kopiert. Nun läßt sich unser Unterprogramm P2 entweder eintippen (zuvor NEW ausführen) oder, wenn es auf Band gespeichert war, mit LOAD laden. Damit steht P1 oberhalb von RAMTOP und P2 im Programmspeicher.

Um beide Programme zu verknüpfen, ist jetzt das zweite Maschinenprogramm mit RAND USR 32743 aufzurufen. Zuvor sollte man den ZX 81 aber unbedingt in den FAST-Modus bringen. Das Maschinenprogramm nutzt nämlich zum Verschieben von P2 eine ROM-Routine (Adresse: 2462d). Dabei zeigt die Systemvariable D-FILE zeitweise nicht auf den Anfang des Bildspeichers, und das kann den ZX 81 beim Bildaufbau durcheinanderbringen (Systemabsturz), sofern er im SLOW-Modus ist.

Da das zweite Maschinenprogramm auch noch P1 vor P2 kopiert, stehen nach Abschluß der Routine und Eingabe von LIST unsere beiden Programmzeilen friedlich untereinander am Bildschirm, Zeile 20 wurde an Zeile 10 angehängt: die Aufgabe ist gelöst.

So richtig nützlich ist die Anknüpfung eines Programms freilich nur bei umfangreicheren Ergänzungen. Dann muß man darauf achten, daß es bei den Zeilennummern keine Überschneidungen gibt. Am besten wird das Programm P2 deshalb vor dem Anknüpfen so umnumeriert, daß es möglichst nahtlos an P1 anschließt.

Wiedergewinnen der vollen RAM-Kapazität

Speicherplatzprobleme kann es geben, wenn man zwei Mammutprogramme miteinander verknüpfen möchte. Dann läßt sich der Bedarf an Speicherplatz am niedrigsten halten, wenn das kürzere der zu verknüpfenden Programme über RAMTOP gesetzt wird. Denn schließlich geht die nach vollzogener Verknüpfung überflüssige Kopie von P1 oberhalb von RAMTOP dem Programmspeicher verloren.

Abhilfe ist möglich, wenn das verknüpfte Gesamtprogramm auf Band gespeichert wird, wir den ZX 81 kurz ausschalten, und dann das Programm wieder laden. RAMTOP hat dann seinen Normalwert.

Da der Umgang mit dem „Brückenbauer" anfangs gewiß gewöhnungsbedürftig ist, gibt die folgende Liste einen zusammenfassenden Überblick auf die nötigen Arbeitsschritte:

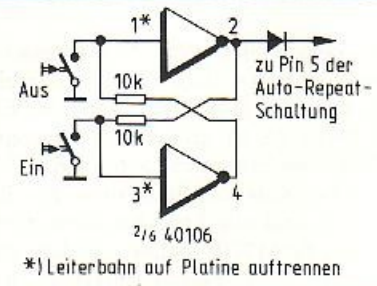
- ○ Länge des Hauptprogramms (P1) bestimmen.
- ○ Append 1 laden und Speicherbedarf eingeben.
- ○ Append 2 laden und starten.
- ○ Hauptprogramm (P1) laden.
- ○ Erstes Maschinenprogramm von Append 2 mit RAND USR 32719 starten.
- ○ Unterprogramm (P2) eventuell umnumeriert laden.
- ○ Zweites Maschinenprogramm von Append 2 mit RAND USR 32743 (im FAST-Modus!) starten.

Andreas Brecht/-ll

Verbesserungsvorschlag:

Flipflop ersetzt Schalter

Zu dem Beitrag über die Auto-Repeat-Schaltung (S. 16) für den ZX 81 haben wir von einem aufmerksamen Leser gleich zwei Anmerkungen bekommen: Zum einen wäre der Ersatztyp für das IC 40106 ein 4584, zum andern lassen sich die übriggebliebenen Schmitt-Trigger dieses ICs noch gut verwenden. Hat man nämlich auf einer externen Tastatur noch zwei Tasten frei, dann können diese die Auto-Repeat-Funktion aktivieren oder stillegen (*Bild*) – der zusätzliche Schalter darf dann entfallen. Wichtig ist, daß bei dieser Lösung die Leiterbahnen an Pin 1 und 3 des ICs aufgetrennt werden, damit diese Eingänge nicht zwangsweise H-Pegel (+5 V) führen.



Flipflop: Diese Schaltung ersetzt bei externen Tastaturen einen Schalter

Renumber-Routine:

Zeilen-Verwalter

Ein kurzes Maschinenprogramm hilft aus der Klemme, wenn in ein eng numeriertes Programm Zeilen einzufügen sind.

Das hier beschriebene Maschinenprogramm wird am besten vor der Entwicklung eines neuen Basic-Programms in eine 26 Zeichen langen REM-Zeile am Progammanfang untergebracht (siehe Eingabeprogramm Seite 72). Jedesmal dann, wenn einen im Basic-Programm zu eng gesetzte Zeilennummern in Bedrängnis bringen, läßt es sich mit RAND USR 16514 starten und schon hat man Platz für Programmeinschübe.

Wer die beiden FUNKSCHAU-Sonderhefte „Zaubern mit dem ZX 81" oder „Klartext für den ZX 81" kennt, der weiß bereits aus Teil 1 der Klartext-Serie, daß der ZX 81 in den ersten beiden Bytes jeder Programmzeile die Zeilennummer unterbringt. Ungewöhnlich ist daran, daß diese Bytes in einer Reihenfolge erscheinen, die für Maschinensprache nicht üblich ist: Das mehr signifikante Byte steht voran! In einer für Maschinensprache üblichen Form ist dagegen die Zeilenlänge, in den nächsten beiden Bytes, gespeichert. Die Zeile

10 REM ABCDEF

hat demnach eine Länge von acht Byte: Ein Byte fürs Schlüsselwort REM, sechs Byte für die Buchstaben ABCDEF sowie ein Byte für den abschließenden NEWLINE-Code.

Mit diesem Grundwissen können wir uns nun der Renumber-Routine (*Bild*) zuwenden. Sie soll im Grundzustand alle Programmzeilen mit der Schrittweite 10 durchnumerieren. Dazu wird das bc-Registerpaar mit Ah (10d) geladen (Adresse 4088h im Listing). Von den anderen beiden Registerpaaren erhält das hl-Registerpaar die Zeilennummer der laufenden Programmzeile zugewiesen. Unter laufender Programmzeile ist diejenige Zeile zu verstehen, die gerade neunumeriert wird. Zu Beginn soll dieser Wert 0000h betragen (4082h). Das de-Registerpaar erhält nun die Adresse 407Dh, das ist der Anfang der ersten Programmzeile (4085h).

Nach Vergabe der Startwerte addiert ein 16-Bit-Arithmetikbefehl die Zahl 10 zur laufenden Programmzeilennummer im hl-Registerpaar (408Bh) und *push hl* wirft diesen Wert auf den Stapel (408Ch). Danach werden die Inhalte der beiden Registerpaare de und hl vertauscht (408Dh), so daß das hl-Registerpaar nunmehr die Adresse des ersten Bytes der laufenden Zeile enthält. Einfacher Datentransport (408Eh und 4090h) kann so die neue Zeilennummer im de-Registerpaar an ihren endgültigen Bestimmungsort bringen, womit die erste Programmzeile neunumeriert ist. Anschließend bringt *dec hl* das hl-Registerpaar wieder auf den Anfang dieser Zeile und das SUCHPROGRAMM bei Adresse 9F2h im ROM sucht nach dem Anfang der nächsten Programmzeile oder dem Beginn der nächsten Variablen (4092h). Im Verlauf dieses Unterprogramms erhält das de-Registerpaar das Ergebnis in Form einer Adresse.

Bevor ein Rücksprung nach Basic zu erwägen ist, muß der ursprüngliche Wert des hl-Registerpaars wieder vom Stapel abgehoben werden. Ein Rücksprung muß genau dann erfolgen, wenn die letzte Programmzeile neunumeriert wurde. Das ist dann der Fall, wenn das Suchprogramm eine Adresse lieferte, deren Inhalt ein NEWLINE-Code ist. Im Listing ist diese Abfrage im Teil zwischen den Adressen 4096h und 409Bh enthalten.

Wie ermittelt das SUCHPROGRAMM aber den Beginn der nächsten Programmzeile? Im hl-Registerpaar steht bekanntlich die Anfangsadresse einer Programmzeile. Um den Anfang der nächsten Zeile zu erhalten, braucht bloß die Zeilenlänge plus 4 addiert werden. Folgendes Maschinenprogramm im ROM führt dies aus:

inc hl	*ld b, (hl)*
inc hl	*inc hl*
ld c, (hl)	*add hl, bc*
inc hl	*ex de, hl*

Das erklärt dann auch die Rücksprungbedienung von vorhin: Würde nämlich die Länge der letzten Programmzeile plus 4 zu deren Anfangsadresse addiert, so stößt man auf den Beginn des Variablenbereichs: Und dieser beginnt mit einem NEWLINE-Code! Im Sinclair-ROM spielt sich das alles zwischen den Adressen 2546 (9F2h) und 2590 (A1Eh) – natürlich etwas komplizierter – ab.

Klaus Herklotz

ADRESSE	BYTES				LABEL	Z-80-ASSEMBLER
4082	21	00	00			ld hl,0000
4085	11	7D	40			ld de,407D
4088	01	0A	00		LOOP	ld bc,000A
408B	09					add hl,bc
408C	E5					push hl
408D	EB					ex de,hl
408E	72					ld (hl),d
408F	23					inc hl
4090	73					ld (hl),e
4091	2B					dec hl
4092	CD	F2	09			call SUCHPRG
4095	E1					pop hl
4096	1A					ld a,(de)
4097	FE	76				cp 76
409A	20	ED				jr nz,LOOP
409B	C9					ret

Renumber-Routine: In der gezeigten Form numeriert sie Programmzeilen mit der Schrittweite 10 allerdings ohne die Sprungziele von GOTO- und GOSUB-Befehlen anzupassen

Programmierbare Lichtorgel:

Licht im Takt

Die hier vorgestellte Lichtorgel bietet, sofern ein 16-KByte-RAM zur Verfügung steht, 15 verschiedene Programme mit jeweils bis zu 50 frei programmierbaren Lichtwechseln. Da werden Ihre Party-Gäste staunen.

Die hier beschriebene programmierbare 4-Kanal-Lichtorgel mit dem ZX 81 wurde als Bühnenlichtsteuerung für Tanzkapellen entwickelt. Herkömmliche Lichtsteuergeräte liefern entweder fest vorgegebene Lichtfolgen oder frequenz- bzw. lautstärkeabhängige Lichtkombinationen.

Oft ist es jedoch wünschenswert, für ein spezielles Musikstück eine individuelle Lichtsequenz zu haben, wobei bestimmte Passagen, die sich nicht unbedingt durch Lautstärke auszeichnen, lichttechnisch hervorgehoben werden.

Zum Bau einer hierzu benötigten programmierbaren Lichtorgel eignet sich der ZX 81 vorzüglich. Mit der 16 KByte RAM-Speichererweiterung können bis zu 15 Lichtsequenzen mit jeweils bis zu 50 Lichtwechseln erstellt, abgespeichert und abgerufen werden.

Der zusätzliche Hardware-Aufwand zur Ansteuerung der Lampen ist minimal, so daß neben Musikern auch Musik-Enthusiasten auf ihre Kosten kommen, die schlicht und einfach in ihren vier Wänden das „optimale Licht" zum bevorzugten Musiktitel genießen möchten.

Benutzerkomfort wird groß geschrieben

Das Programm startet nach dem Laden automatisch. Bei erstmaligem Gebrauch wird zunächst nach der Anzahl der zu erstellenden Lichtfolgen und der maximalen Zahl der Lichtwechsel pro Lichtfolge gefragt. Nach diesen Angaben richtet sich später die Speicher- und Ladezeit des Programms.

Wurden bereits eine oder mehr Lichtfolgen programmiert, meldet sich das Programm direkt mit dem Menü, das die Wahl bietet zwischen Eingeben, Starten und Abspeichern. Vor der Eingabe wird abgefragt, wie oft innerhalb eines Taktes ein Lichtwechsel möglich sein soll.

Wird beispielsweise „Lichtwechsel jeden ¼ Takt" gewählt, ergibt dies viermal pro Takt die Möglichkeit, eine Lichtkombination einzugeben. Die Wahl, welche Lampen leuchten sollen, erfolgt über die Tasten 1, 2, 3 und 4, die dem jeweiligen Kanal entsprechen.

Die Eingabe wird jeweils mit der Taste 0 abgeschlossen. Es bestehen verschiedene Editier-Möglichkeiten, die durch das Programm selbst beschrieben werden.

Vor dem Starten ist durch viermaliges Drücken der Taste 0 die Taktdauer einzugeben (ähnlich dem Einzählen eines Schlagzeugers). Dabei ist die Taste nur kurz zu berühren. Danach wird die gewünschte Lichtfolge durch Eingabe der entsprechenden Nummer abgerufen.

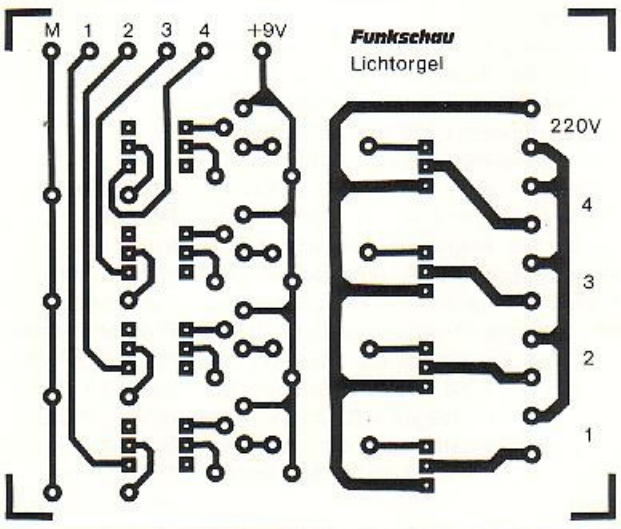
Die Abfolge des Programms taucht dann zur Kontrolle am Bildschirm auf. Schnelle Lichtsequenzen führen aber dazu, daß, bedingt durch die für die Bilddarstellung benötigte Zeit, die Lichtausgabe nicht taktsynchron erfolgen kann. Dieser Zustand wird angezeigt; gleichzeitig steht es einem jedoch frei, die Bildkontrolle abzuschalten, um die Synchronität zu erreichen.

Die Lichtfolge wiederholt sich, bis sie unterbrochen (Taste P) oder abgebrochen (Taste S) wird. Da sich ZX-81-Rechner untereinander in der Rechengeschwindigkeit unterscheiden, ist unter Umständen ein Abgleich der Taktdauer vorzunehmen. Dazu ist die Variable U in Zeile 75 abzuändern.

Zur Steuerung genügt ein Ausgabe-Interface

Eine 4-Kanal-Lichtorgel bietet 16 verschiedene Lichtkombinationen. Weist man den angeschalteten Kanälen logisch 1 und den nicht angeschalteten Kanälen logisch 0 zu, erhält man für jede Kombination eine Dezimalzahl zwischen 0 und 15. Sollen zum Beispiel Kanal 1 und Kanal 3 angeschaltet werden, lautet die Zahl 5, gemäß der Zuordnung in der *Tabelle*.

Die Dezimalzahlen werden abgespeichert und bei der Ausgabe mit dem Befehl POKE 49185, Zahl auf den Datenbus gelegt. Damit liegt der Binärwert der

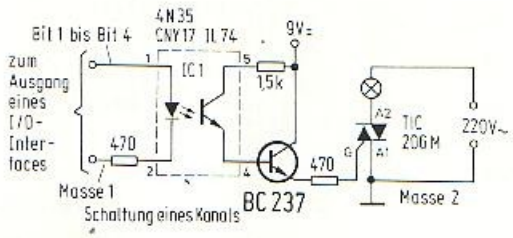


① **Platinenlayout:** Im Betrieb liegt an einigen Leiterbahnen Netzspannung; dann heißt es: Finger weg!

Zahl am Out-Port des zur Steuerung nötigen Interfaces.

Das Interface ist die Schnittstelle zwischen dem ZX 81 und der Hardware der Lichtorgel (Lampenansteuerung). Ein direkter Anschluß der Lampenansteuerung an den Erweiterungssport des ZX 81 ist nicht möglich, weil die Steuerdaten nur sehr kurz auf dem Datenbus liegen und ohne Adreßdecoder keine Verknüpfung zwischen den Steuerdaten und der Adresse besteht, unter denen sie erteilt werden.

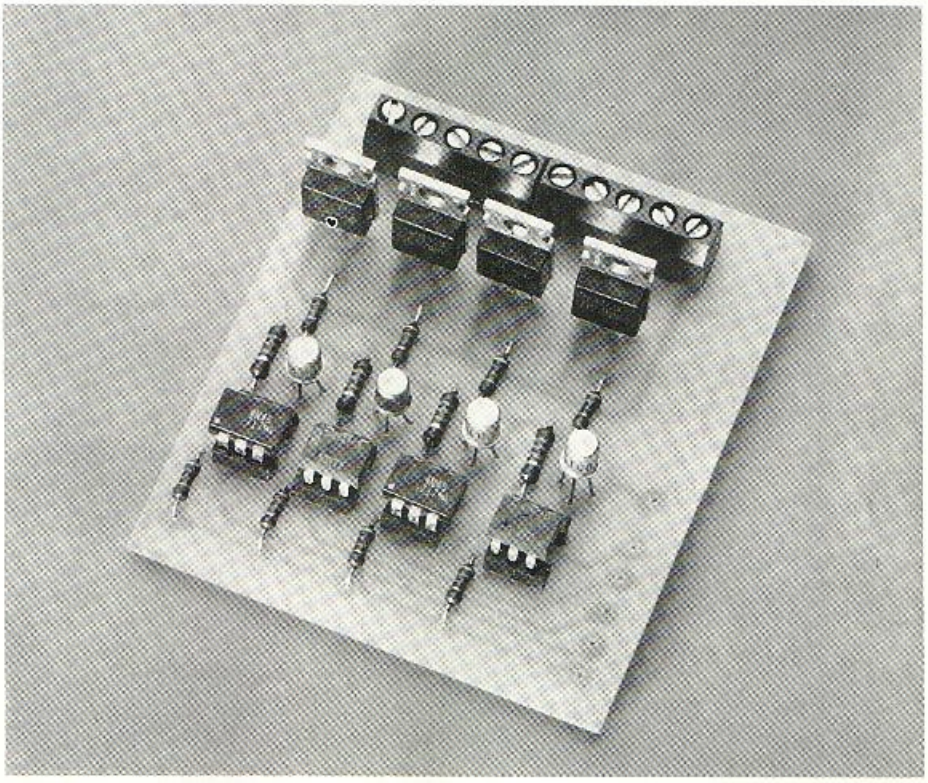
Aufgabe des Interfaces ist es daher, die Steuerdaten, die unter der „Portadresse" (hier z. B. 49185) eingetroffen sind, so-



② **Lampenansteuerung:** Ein Optokoppler übernimmt die galvanische Trennung. Das Steuersignal, auch für die übrigen drei identischen Kanäle, liefert ein I/O-Interface

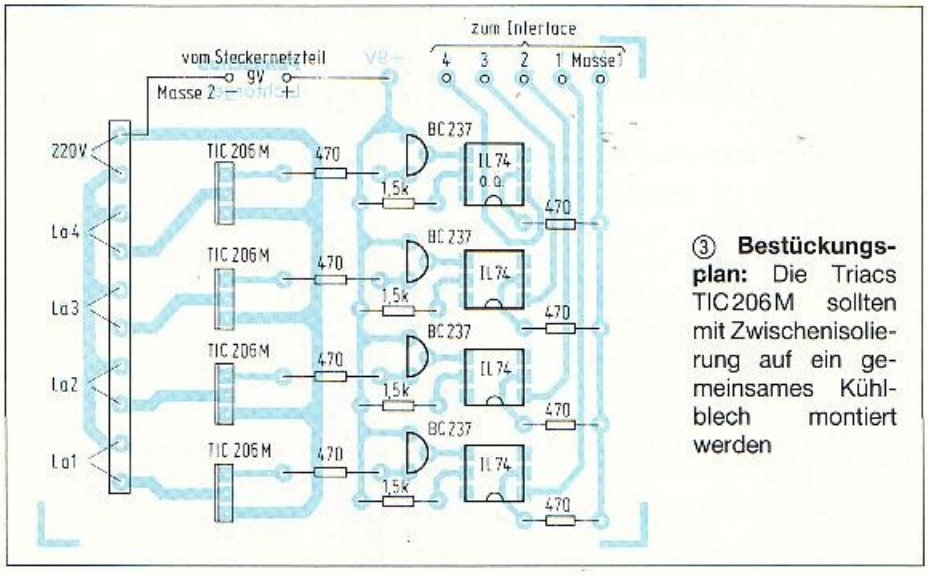
Tabelle: Die 4-Kanal-Lichtorgel bietet 16 Lichtkombinationen, denen im Programm die „Codenummern" 0 bis 15 zugewiesen werden. 1: Lampe leuchtet, 0: Lampe leuchtet nicht

Zahl	Kanäle			
	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1



Triacsteuerung: Diese Platine muß samt Spannungsversorgung unbedingt berührungssicher montiert werden

Foto: Neumayr



③ **Bestückungsplan:** Die Triacs TIC206M sollten mit Zwischenisolierung auf ein gemeinsames Kühlblech montiert werden

lange an die Lampensteuerung weiterzugeben, bis neue Steuerdaten eintreffen. Dafür läßt sich jedes beliebige I/O-Parallel-Interface verwenden, z. B. die Basisplatine der „à-la-carte"-Serie oder der PIO-Port, der ab Seite 18 beschrieben ist. Da das Interface jedoch nur die Ausgabe von Daten beherrschen muß und keine Dateneingabe, ist das ab Seite 70 beschriebene Spar-Interface die einfachste und billigste Lösung. Die vom Programm angesprochene Portadresse ist auf dieses Interface zugeschnitten.

Wer ein anderes Interface benutzen möchte, muß dessen Portadresse anstelle der jetzigen in Programmzeile 662 eintragen. Beim Betrieb mit dem PIO-Port ist in einer Programmzeile 1 das Maschinenprogramm zur Datenausgabe einzu-

```
   5 LET V=0
  10 CLS
  11 PRINT "WIEVIELE LICHTFOLGEN
 WOLLEN"
  12 PRINT "SIE ERSTELLEN ? (<=1
5)"
  14 INPUT Q
  15 IF Q>15 THEN GOTO 14
  16 PRINT
  17 PRINT "WIEVIELE LICHTWECHSE
L PRO"
  18 PRINT "LICHTFOLGE MAXIMAL ?
 (<=50)"
  19 INPUT C
  20 IF C>50 THEN GOTO 19
  25 DIM B(C)
  26 DIM S(C)
  29 DIM A(Q,C)
  30 DIM W(4)
  32 DIM U(C)
  33 DIM T(C)
  34 LET E6=0
  35 LET E8=0
  39 LET Z=0
  40 LET L$="***"
  60 LET M$="
    "
  70 CLS
  75 LET U=10
  80 PRINT "ZX 81 ALS PROGRAMMIE
RBARE"
  85 PRINT "          LICHTORGEL"
  90 PRINT
  95 PRINT
 100 PRINT "       STARTEN  -> 1"
 105 PRINT
 110 PRINT "       EINGEBEN -> 2"
 115 PRINT
 120 PRINT "       SPEICHERN-> 3"

 125 LET H8=0
 126 LET H9=0
 127 LET E9=0
 130 LET Q$=INKEY$
 140 IF Q$="" THEN GOTO 130
 150 IF Q$="1" THEN GOTO 170
 155 IF Q$="2" THEN GOTO 1000
 157 IF Q$="3" THEN GOTO 5000
 160 GOTO 130
 170 CLS
 180 IF V=0 THEN GOTO 190
 185 GOTO 200
 190 PRINT "ES WURDE NOCH KEINE
LICHTFOLGE  ERSTELLT ."
 192 PAUSE 100
 195 GOTO 70
 200 LET I1=1
 230 PRINT "BITTE TAKTDAUER EING
EBEN :"
 231 PRINT
 232 PRINT "(4 MAL KURZES DRUECK
EN DER       TASTE 0)"
 240 PRINT
 250 IF INKEY$<>"0" THEN GOTO 25
0
 257 POKE 16436,255
 258 POKE 16437,255
 260 FOR I=1 TO 5
 262 NEXT I
 270 IF INKEY$<>"0" THEN GOTO 27
0
 285 LET W(I1)=(65535-PEEK 16436
-256*PEEK 16437)
 290 LET I1=I1+1
 300 IF I1=4 THEN GOTO 320
 310 GOTO 257
 320 LET W=(W(1)+W(2)+W(3))/3
 400 PRINT "DANKE"
 410 PAUSE 30
 470 CLS
 480 REM **** LICHTFOLGE *******
 490 PRINT "BITTE NUMMER DER LIC
HT-"
 495 PRINT "FOLGE EINGEBEN :"
 497 PRINT "(";
 500 FOR I=1 TO V
 510 PRINT I;",";
 515 NEXT I
 520 PRINT ")"
 530 INPUT R
 540 IF R>V THEN GOTO 530
 600 REM *** ABLAUF **********
 605 GOSUB 610
 607 GOTO 70
 610 CLS
 612 PRINT "LICHTFOLGE NR. ";R
 620 PRINT "STOP  -> S"
 630 PRINT "PAUSE -> P"
 640 IF E9 THEN PRINT AT 7,0;"VE
RBESSERN -> V"
 645 LET W2=((W*4)/S(R))-U
 650 FOR I=1 TO B(R)
 655 POKE 16436,255
 657 POKE 16437,255
 660 IF A(R,I)=16 THEN LET I=1
 662 POKE 49185,A(R,I)
 665 IF H9 THEN GOTO 880
 670 LET A1=0
 680 LET A2=0
 690 LET A3=0
 700 LET A4=0
 710 LET A1=INT (A(R,I)/8)
 720 LET A2=INT ((A(R,I)-8*A1)/4
)
 730 LET A3=INT ((A(R,I)-8*A1-4*
A2)/2)
 740 LET A4=A(R,I)-8*A1-4*A2-2*A
3
 745 PRINT AT 11,1;M$
 750 IF A1 THEN PRINT AT 11,23;L
$
 770 IF A2 THEN PRINT AT 11,16;L
$
 790 IF A3 THEN PRINT AT 11,9;L$
 810 IF A4 THEN PRINT AT 11,2;L$
 880 LET Q$=INKEY$
 890 IF Q$="S" THEN GOTO 70
 900 IF Q$="P" THEN GOTO 930
 910 IF Q$="V" THEN GOSUB 3000
 920 GOTO 950
 930 PRINT AT 15,1;"WEITER -> NL
"
 940 INPUT Z$
 945 PRINT AT 15,1;M$
 947 GOTO 969
 950 LET W1=(65535-PEEK 16436-25
6*PEEK 16437)
 952 IF W1<W2 THEN GOTO 950
 955 IF H8 OR H9 OR E9 THEN GOTO
 969
 956 IF W1>W2+8 THEN GOTO 958
 957 GOTO 969
 958 PRINT "AUSGABE NICHT TAKTSY
NCHRON."
 959 PRINT "BILDKONTROLLE STOPPE
N ?(J/N)"
 960 LET Q$=INKEY$
 961 IF Q$="N" THEN GOTO 967
 962 IF Q$="J" THEN GOTO 964
 963 GOTO 960
 964 LET H9=1
 965 PRINT AT 12,0;M$
 966 GOTO 968
 967 LET H8=1
 968 PRINT AT 13,0;M$
 969 NEXT I
 970 RETURN
1000 REM *** TAKTWAHL *******
1005 LET V=V+1
1006 LET K=1
1007 LET J=1
1008 LET J1=1
1010 CLS
1015 PRINT "WUENSCHEN SIE"
1020 PRINT "LICHTWECHSEL JEDEN"
1025 PRINT
1030 PRINT "       TAKT  -> 1"
1040 PRINT "  1/2 TAKT  -> 2"
1050 PRINT "  1/4 TAKT  -> 4"
1060 PRINT "  1/8 TAKT  -> 8 ?"
1070 LET Q$=INKEY$
1075 IF Q$="" THEN GOTO 1070
1080 LET Q=VAL Q$
1090 IF Q<1 OR Q=3 OR Q=5 OR Q=6
 OR Q=7 OR Q>8 THEN GOTO 1070
1100 LET T=Q
1105 LET S(V)=T
1110 CLS
1115 PRINT "LICHTFOLGE NR.";V
1117 PRINT
1120 PRINT "BITTE KANAELE EINGEB
EN :"
1130 PRINT "(TASTEN 1,2,3 UND 4)
"
1140 PRINT AT 12,3;"1";"      ";
"2";"      ";"3";"      ";"4"
1150 PRINT AT 14,0;"EINGABE    -
> 0"
1160 PRINT "VERBESSERN -> V"
1170 PRINT "KONTROLLE  -> K"
1175 PRINT "ENDE       -> E"
1177 LET T(J)=K
1178 LET U(J)=J1
1180 IF T=1 THEN LET T(J)=J
1182 LET E8=0
1185 PRINT AT 20,0;T(J);".TAKT"
1194 LET J2=J1
1196 LET J3=K
1210 IF T=2 THEN GOTO 1212
1211 GOTO 1220
1212 PRINT AT 20,14;U(J);".HALBT
AKT"
1213 LET Z=Z+1
1220 IF T=4 THEN GOTO 1222
1221 GOTO 1230
1222 PRINT AT 20,14;U(J);".VIERT
ELTAKT"
1223 LET Z=Z+1
1230 IF T=8 THEN GOTO 1232
1231 GOTO 1237
1232 PRINT AT 20,14;U(J);".ACHTE
LTAKT"
1233 LET Z=Z+1
1237 IF Z=T THEN GOSUB 1420
1260 PRINT AT 11,0;M$
1270 LET A1=0
1271 LET A2=0
1272 LET A3=0
1273 LET A4=0
1280 LET Q$=INKEY$
1285 IF Q$="" THEN GOTO 1280
1290 IF Q$="1" THEN GOTO 1292
1291 GOTO 1300
1292 PRINT AT 11,2;L$
1293 LET A1=1
1300 IF Q$="2" THEN GOTO 1302
1301 GOTO 1310
1302 PRINT AT 11,9;L$
1303 LET A2=1
1310 IF Q$="3" THEN GOTO 1312
1311 GOTO 1320
1312 PRINT AT 11,16;L$
1313 LET A3=1
1320 IF Q$="4" THEN GOTO 1322
1321 GOTO 1330
1322 PRINT AT 11,23;L$
1323 LET A4=1
1330 IF Q$="K" THEN GOTO 1440
1340 IF Q$="E" THEN GOTO 1342
1341 GOTO 1350
1342 LET A(V,J)=16
1343 LET B(V)=J
1345 GOTO 32
1350 IF Q$="V" THEN GOTO 1352
1351 GOTO 1360
1352 PRINT AT 11,0;M$
1353 GOTO 1270
1360 IF Q$<>"0" THEN GOTO 1280
1370 LET X=A1+A2*2+A3*4+A4*8
1375 PRINT AT 11,0;M$
1380 LET A(V,J)=X
1385 IF E6 THEN GOTO 3050
1390 IF E6=0 THEN LET B(V)=J
1395 LET E6=0
1410 LET J=J+1
1412 LET J1=J1+1
1415 GOTO 1177
1420 IF E6 THEN RETURN
1421 LET Z=0
1422 LET J1=0
1423 LET K=K+1
1424 LET E8=1
1425 RETURN
1430 PRINT AT 7,0;M$
1440 LET W=0
1441 LET R=V
1442 LET E9=1
1445 LET B(V)=J
1450 IF E8 THEN GOTO 1458
1455 GOTO 1465
1458 LET J1=J2
1459 LET K=J3
1460 LET Z=T
1465 LET Z=Z-1
1466 GOSUB 610
1470 PRINT AT 7,0;M$
1480 GOTO 1110
3000 LET E6=1
3010 PRINT AT 7,0;"BITTE NEU EIN
GEBEN :"
3017 PRINT AT 12,3;"1";"      ";
"2";"      ";"3";"      ";"4"
3020 LET J8=J
3025 IF T=1 THEN GOTO 3035
3030 LET J=(T(I)-1)*T+U(I)
3032 GOTO 1180
3035 LET J=T(I)
3040 GOTO 1180
3050 LET J=J8
3052 LET Z=Z-1
3055 PRINT AT 7,0;M$
3060 PRINT AT 12,1;M$
3065 PRINT AT 20,0;M$
3067 LET E6=0
3070 RETURN
5000 REM *** ABSPEICHERN ***
5010 CLS
5020 PRINT "BEREIT -> NEWLINE"
5025 INPUT X$
5030 SAVE "LICHTORGE1"
5040 IF V=0 THEN RUN
5050 GOTO 32
```

Programmlisting Lichtorgel: Eine gründliche Benutzerführung hilft bei der Eingabe der gewünschten Lichtsequenzen

geben (siehe Seite 23) und der POKE-Befehl zur Datenausgabe ähnlich zu verändern, wie in Teil 9 der „à-la-carte"-Serie beschrieben (siehe Seite 32). Von den Datenausgängen am Interface werden nur die vier niederwertigen benutzt.

Optokoppler halten dem ZX 81 die 220 V vom Leib

Die Schaltung der Lampenansteuerung zeigt *Bild 2*. Eine galvanische Trennung zwischen Interface und Lichtnetz erfolgt mit Hilfe von Optokopplern (IC1). Bei logisch 1 am Eingang leuchtet die Leuchtdiode und der Ausgangs-Fototransistor schaltet durch. Er steuert einen weiteren Transistor an, der den Zündstrom für den Triac liefert. Der Triac schaltet dann das Lichtnetz an die einzelnen Lampen des entsprechenden Kanals.

Zur besseren Wärmeabfuhr sollten die Triacs auf Kühlbleche montiert werden. Die maximale Belastbarkeit pro Triac beträgt 1600 W; pro Kanal lassen sich daher mehrere Lampen anschließen. Die *Bilder 1* und *3* zeigen das Platinenlayout und den Bestückungsplan.

Die Spannungsversorgung der Schaltung (ca. 9 V) darf aus Sicherheitsgründen nicht aus dem Computernetzteil erfolgen, sondern entweder über Batterie oder aus einer eigenen Versorgungsschaltung. Am besten wird ein schutzisoliertes Steckernetzteil verwendet, auf keinen Fall ein übliches Netzgerät mit geerdeter Massebuchse. Der Grund für das Verbot: Masse 2 kann Netzspannung führen! Daher muß das gesamte Netzteil berührungssicher sein, denn es kann auch sekundärseitig 220 V gegen Erde führen. Beim Betrieb der Lichtorgel sind unbedingt die üblichen Vorsichtsmaßnahmen im Umgang mit netzspannungsführenden Schaltungen zu beachten (Lebensgefahr!).

Läuft alles zur Zufriedenheit, ist die gesamte Schaltung der Lampenansteuerung berührungssicher, am besten in einem Kunststoffgehäuse, unterzubringen. Mit Bohrungen ist dann für eine Luftzirkulation zu sorgen, damit es im Gehäuse nicht zu einem Wärmestau kommt.

Klaus Stuchlich

Laufschrift:

Textausgabe am laufenden Band

ZX 81, Fernsehgerät und Software – diese Zutaten reichen schon aus, um ein interesseweckendes Informationssystem aufzubauen. Das Programm zeigt auf dem Bildschirm eine Laufschriftzeile mit frei wählbarem Text (fast) beliebiger Länge.

Viele Computerdarstellungen auf dem Bildschirm ließen sich interessanter gestalten, wenn man zusätzlich einen durchlaufenden Schriftzug, ähnlich wie bei den Laufschriftdisplays, darstellen könnte. Abgesehen vom höheren Aufmerksamkeitswert braucht eine solche Darstellungsart unabhängig von der Länge des Textes nur den Platz einer einzigen Bildschirmzeile. Das ist gerade dann von Vorteil, wenn zusätzlich andere Informationen (z. B. Grafiken, Meßdaten, Uhrzeit) ständig auf dem Bildschirm dargestellt werden.

Mit Hilfe des Sinclair-Basic läßt sich das Programm einfach gestalten. Der Programmteil bis Zeile 50 dient dazu, den Text der Laufschrift in den ZX 81 zu übernehmen (FAST beschleunigt die Eingabe von längeren Texten). Ans Ende des Textes hängt Programmzeile 80 zur Kennzeichnung noch zwei Sternchen an. Ist der Text kürzer als die 32 Zeichen einer Bildschirmzeile, so füllen die Programmzeilen 90 bis 110 die Lücke mit Leerzeichen auf.

Läuft der Text erst einmal über den Bildschirm, so läßt er sich durch Druck auf das Tastenfeld „A" anhalten (Programmzeile 150) oder über Tastenfeld „S" auf Magnetband abspeichern (Programmzeile 160).

```
 10 REM LAUFSCHRIFT
 20 FAST
 30 LET N=1
 40 PRINT "GEBEN SIE BITTE DEN
TEXT EIN"
 50 INPUT A$
 60 CLS
 70 SLOW
 80 LET A$=A$+" ** "
 90 FOR X=1 TO 32-LEN A$
100 LET A$=A$+" "
110 NEXT X
120 LET A$=A$+A$( TO 32)
130 PRINT AT 21,0;A$(N TO 31+N)
140 LET N=N+1
150 IF INKEY$="A" THEN GOTO 150
160 IF INKEY$="S" THEN GOTO 200
170 IF N=LEN A$-32 THEN LET N=1
180 GOTO 130
200 POKE 16389,68
210 SAVE "LAUFSCHRIFT"
220 GOTO 130
```

Listing: Endlos jagt dieses Programm eine Laufschrift über den Bildschirm

Das Programm selbst bietet Anreiz zur Weiterentwicklung. So ist die Laufschrift auch mit Grafik zu koppeln, wie es der Ausdruck z. B. am Franzis-Signet zeigt. Der Text ließe sich auch statt über INPUT A$ (Programmzeile 50) mit LET A$ = „Laufschrift..." eingeben. Auch ließen sich gleichzeitig mehrere Texte abspeichern und abrufen z. B. mit LET A$(1)="...", LET A$(2)="...".

Wolf-Dieter Roth

ZX-81-Hardwaretip:

Wissenswertes zum RAM-Modul

Auf Seite 48 haben wir ausführlich über die Probleme berichtet, die die unvollständige Adreßdecodierung des ROMs im ZX 81 bereitet. Ohne zusätzlichen Adreßdecoder ist das ROM aber auch schon allein durch Anschließen eines RAM-Moduls vollständig decodiert!

Probleme macht bei den RAM-Modulen von Memotech die nicht durchgeschleifte Adreßleitung A 14, und zwar dann, wenn Hardware hinter dem Modul angesteckt wird. Abhilfe: Die beiden äußeren Codierschalter am Modul auf On (oben) stellen, die beiden mittleren auf Off. Mit dieser Schalterstellung lassen sich allerdings nicht mehr zwei 16-KByte-RAMs zu einem 32-KByte-RAM zusammenfügen.

Speicherprobleme sind zu erwarten, wenn man die 9-V-Versorgungsspannung des ZX 81 stark reduziert (Verlustwärme). Die bipolaren RAM-ICs der Module brauchen die höhere Spannung. -ll

8-Bit-Datenausgabe:

Spar-Interface

Der Trick mit der „Geisteradresse"

Hier präsentieren wir einen Ausgabe-Port für 8-Bit-Daten, der mit geringstmöglichem Aufwand auskommt. Einzige Einschränkung: Der ZX 81 kann nicht mehr als 16 KByte RAM nutzen.

Wer mit seinem ZX 81 lediglich periphere Geräte ansteuern möchte, z.B. wie auf Seite 66 beschrieben eine Lampenkette (oder den Telefoncomputer von Seite 33), wer Steueraufgaben automatisieren oder den Computer als Musikinstrument benutzen möchte, der hat gewiß schon mit dem Gedanken gespielt, sich ein Ausgabe-Interface zuzulegen oder den Selbstbau in Angriff zu nehmen.

Die im Laufe der Zeit veröffentlichten Schaltungen solcher Interfaces verfügen stets über eine Baugruppe, die ziemlich aufwendig eine Decodierung der vom ZX 81 bereitgestellten Adreßsignale vornimmt. Die damit verbundene Beschaltung der 16 Adreßleitungen hat sicherlich manchen Hobbyelektroniker abgeschreckt.

Adressen über 32767 haben eine Sonderstellung

Wer seinen ZX 81 mit nicht mehr als 16 KByte RAM ausgestattet hat, verfügt über die dann eigentlich brachliegende Adreßleitung A15.

Machen wir uns dazu folgendes klar: Bei einem 16-KByte-RAM ist Adresse 32767 die höchste nutzbare Adresse. In der Binärdarstellung dieser Adresse

0111 1111 1111 1111

erkennen wir, daß – von rechts gelesen – nur die Adreßleitungen A0...A14 angesprochen werden (logisch 1). Adreßleitung A15 geht erst dann auf logisch 1, wenn Adressen über 32 767 angesprochen werden. Das aber sind Adressen, die, wenn unser ZX 81 mit nicht mehr als 16 KByte RAM ausgestattet ist, gar keine RAM-Speicherzelle mehr ansprechen können.

Dennoch können wir eine so hohe Adresse verwenden. Der Z-80-Mikroprozessor weist ihr unbeirrbar Daten zu, ganz gleich, ob der adressierte Speicherplatz vorhanden ist oder nicht. Wenn wir z. B. POKE 49185,1 eingeben, dann liegt für einen kurzen Augenblick die Adresse 49185 in der Binärform

1100 0000 0010 0001

auf den Adreßleitungen A0...A15. Gleichzeitig liegt auf den Datenleitungen D0...D7 das binäre Äquivalent unserer Zahl 1 an, also:

0000 0001

Ohne weitere Maßnahmen hat das „Poken" eines nicht vorhandenen Speicherplatzes selbstverständlich keinen Sinn. Unser kleines Interface können wir auf diese Weise jedoch ansprechen. Der große Vorteil liegt darin, daß nicht alle 16 Adreßleitungen abgefragt werden müssen, sondern nur die Adreßleitung A15.

Der Adreßdecoder wertet nur drei Signale aus

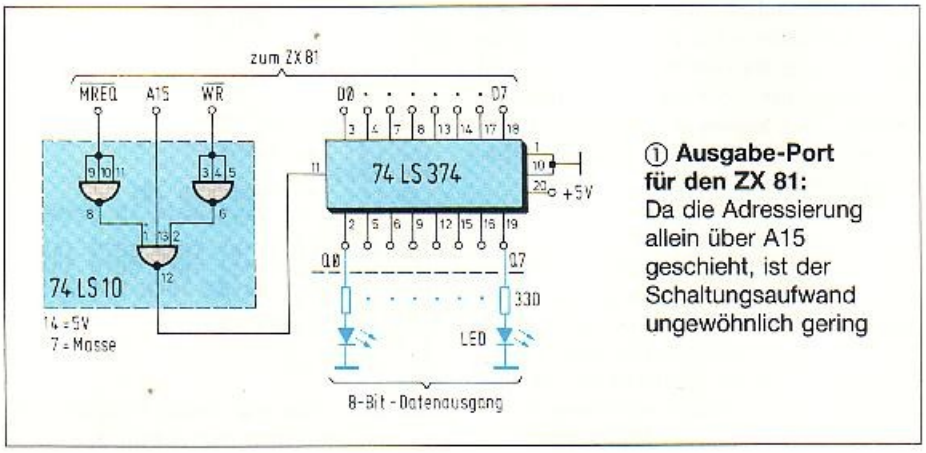
Bei der Abfrage gilt es jedoch sicherzustellen, daß unsere Adreßleitung vom ZX 81 im Rahmen seines Betriebssystems nicht gerade anderweitig benutzt wird. Der Datenbus muß also genau in dem Moment „abgefragt" werden, in dem der Mikroprozessor einen adressierten Speicherplatz auffordert, Daten zu übernehmen.

Beim Mikroprozessor des ZX 81 geschieht dies dadurch, daß die Leitungen $\overline{WR}$ (Write) und $\overline{MREQ}$ (Memory Request) kurzzeitig L-Pegel (logisch 0) führen. Durch einen Vergleich ist sicherzustellen, daß erst unter dieser Voraussetzung H-Pegel auf A15 für unser Interface gültig ist.

Ein 74LS10 (drei NAND-Glieder mit je drei Eingängen) übernimmt diese Aufgabe (*Bild 1*). Nachdem er die Signale $\overline{MREQ}$ und $\overline{WR}$ invertiert und sie mit dem Signal auf der Adreßleitung A15 verglichen hat (UND-Verknüpfung), stellt er uns, gesetzt den Fall, wir haben eine Adresse größer 32767 „gepoked", kurz L-Pegel an Pin 12 zur Verfügung. Diesen nutzen wir auf der Stelle, um mit Hilfe eines Speicher-ICs 74LS374 die im selben Moment auf dem Datenbus liegende Information zu sichern.

Sorglos adressieren mit „Geisteradressen"

Aber Vorsicht beim Betrieb ohne RAM-Erweiterung! Der unvollständigen Adreßdecodierung des ZX 81 wegen



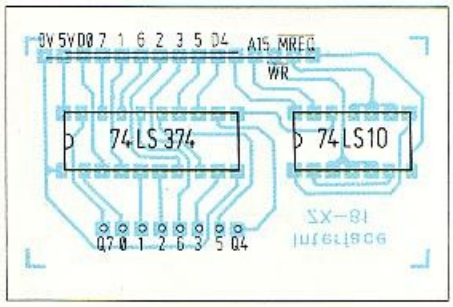
① Ausgabe-Port für den ZX 81: Da die Adressierung allein über A15 geschieht, ist der Schaltungsaufwand ungewöhnlich gering

sprechen wir dann beim Adressieren einer nicht vorhandenen Speicherstelle gleichzeitig eine vorhandene Speicherstelle an (siehe auch Seite 48). Im Fall der Adresse 49185 ist dies der Speicherplatz 16417. Deutlich wird das, wenn wir die binäre Schreibweise dieser Adressen betrachten.

49185: 1100 0000 0010 0001
16417: 0100 0000 0010 0001

Einen Unterschied gibt es lediglich bei A15. Ohne RAM-Erweiterung erkennt der ZX 81 diesen Unterschied jedoch nicht und hält die beiden Adressen nicht auseinander.

Unvorsichtiges Poken nicht vorhandener Speicherplätze kann deshalb zu Störungen eines Programmablaufs bis hin zum „Absturz" des ZX 81 führen. Die im Beispiel gewählte Adresse 49185 birgt diese Gefahr indes nicht! Ihr Duplikat 16417 liegt im Bereich der Systemvariablen und wird vom ZX 81 nicht benutzt (siehe Sinclair-Handbuch, Kapitel über Systemvariablen). Genauso verhält es sich mit der Adresse 16507, die wir erreichen, wenn wir POKE 49275,X eingeben.



② **Platinenlayout:** Die einfache Schnittstelle läßt sich sogar noch im Gehäuse des ZX 81 unterbringen

Damit erweisen sich die „Geisteradressen" 49185 und 49275 als ideal, um unser kleines Interface auch in der Grundversion des ZX 81 risikolos anzusprechen. Wer jetzt in einem Programm eine andere Adresse als die genannten benutzen möchte, muß nun, um die entsprechende Geisteradresse zu finden, nicht jedesmal die Binärdarstellung ermitteln. Es genügt vielmehr 32768 zu der tatsächlich vorhandenen Adresse zu addieren.

Selbstverständlich können wir unser Interface auch von Maschinenprogrammen aus ansprechen (Adresse 49185 hat hexadezimal den Wert C021h).

Wer seinem ZX 81 mehr als 16 KByte RAM spendiert hat, kann ebenfalls mit dem Interface arbeiten. Dann muß man sich aber auf die Nutzung von 16 KByte beschränken (RAMTOP setzen).

Testen läßt sich das Interface am einfachsten, wenn man an die Ausgänge des Speicher-ICs zunächst einmals LEDs anschließt. Das Programm

```
10 LET I=0
20 POKE 49185,I
30 PAUSE 50
40 LET I=255
50 POKE 49185,I
60 PAUSE 50
70 GOTO 10
```

sorgt dann dafür, daß alle LEDs im 1-s-Rhythmus gemeinsam blinken. Wegen des geringen Verdrahtungsaufwandes steht einem Einbau der Schaltung in den ZX 81 nichts im Wege. Das Zwischenschalten von Pufferbausteinen ist nur dann nötig, wenn die Leitungen des Mikroprozessors noch mit weiteren Peripherieschaltungen verbunden werden.

Frank Möcke

ZX-81-Hardwaretip:

Tempo durch Nachbrenner

Beim Zeichnen von Graphiken oder bei längeren Berechnungen wird oft kein Gebrauch vom FAST-Modus des ZX 81 gemacht, um den Ablauf des Programms besser verfolgen zu können. Später, wenn man das Programm kennt, sitzen indes alle ungeduldig zappelnd vor dem Computer und würden ihn am liebsten „anschieben", damit er etwas schneller läuft.

Die Trägheit des ZX 81 im SLOW-Modus beruht darauf, daß seine Zentraleinheit, der Mikroprozessor Z 80 A, nicht nur das Programm abarbeiten, sondern auch noch 50mal in jeder Sekunde ein Fernsehbild aufbauen muß. Dies beansprucht etwa ¾ der Rechenkapazität, wobei der Rechner im SLOW-Modus ständig über den „nicht maskierbaren Interrupt" (NMI), das Programm unterbricht.

Im FAST-Modus kümmert sich der Rechner dagegen nur dann um den Bildschirm, wenn er gerade nichts besseres zu tun hat, z. B. wenn er auf eine Eingabe wartet oder das Programm abgelaufen ist. Um nun nach Belieben in den FAST-Modus umzuschalten, genügt es, das von außen zugängliche NMI-Signal nach +5 V kurzzuschließen. Das Programm wird dann nicht mehr unterbrochen und der Rechner arbeitet mit voller Geschwindigkeit.

Allerdings sollte man von Zeit zu Zeit einmal nachschauen, ob der Rechner vielleicht auf eine Eingabe wartet oder das Programm gar schon abgelaufen ist. Der ZX 81 kann sich nämlich nicht melden solange das NMI-Signal blockiert ist.

Der Umschalter kann über einen Stekker an die Sinclair-Schnittstelle angeschlossen werden, es ist also kein Eingriff in den Rechner nötig. Der Schalter oder Taster – es sollte ein prellfreier Typ sein – wird mit einem möglichst kurzen Kabel zwischen Pin 1 auf der Unterseite (+5 V) und Pin 12 auf der Oberseite ($\overline{\text{NMI}}$) der Schnittstelle angeschlossen. Der Kurzschlußstrom beträgt nur etwa 2 mA, so daß man keine Beschädigung des ZX 81 zu befürchten braucht.

Die Pins der Schnittstelle werden von links nach rechts aufwärts gezählt, wenn man den Rechner „betriebsmäßig" vor sich stehen hat, also mit der Steckerleiste nach hinten. Die Aussparung zählt als Pin 3.

Sobald der ZX 81 einmal softwaremäßig im FAST-Modus ist (dies ist außer nach FAST auch während der Ausführung der Befehle PAUSE, NEW, COPY, SAVE, LOAD und nach dem Einschalten des Rechners der Fall), merkt er den „Betrug" mit dem Kurzschlußschalter und weigert sich wieder in den SLOW-Modus zu gehen, solange der Kurzschluß besteht.

Neben der hardwaremäßigen SLOW/FAST-Umschaltung gibt es auch eine softwaremäßige Lösung, die aber nur fallweise zum Ziel führt. Dazu sind die folgenden Basic-Zeilen in ein Programm einzufügen:

```
IF INKEY $ = "F" THEN FAST
IF INKEY $ = "S" THEN SLOW
```

Läuft das Programm im SLOW-Modus, dann wird beim Drücken der Taste F solange in den FAST-Modus umgeschaltet bis man mit Taste S wieder SLOW wählt. Das klappt selbstverständlich nur dann, wenn die beiden Basic-Zeilen vom Programm auch ständig abgearbeitet werden, also z. B. in einer Schleife stehen.

Wolf-Dieter Roth

Grundlagen:

Eingabe von Maschinencodes

Einige Beiträge in diesem Sonderheft verlangen die Eingabe von Maschinenprogrammen. Wer dabei nicht ganz sattelfest ist, bekommt hier auf die Schnelle ein Kochrezept geboten.

Schreibt man auf dem ZX 81 Basic-Programme, dann muß man sich keine Gedanken darüber machen, wo und wie das Programm im Speicher untergebracht wird. Das wird alles vom Betriebssystem des Computers geregelt. Die Eingabe von Maschinenprogrammen wird dagegen vom Betriebssystem nicht unterstützt.

Um Maschinenprogramme im Speicher des ZX 81 unterzubringen, muß zunächst einmal Platz für sie reserviert werden. Dafür bietet sich die Basic-Anweisung REM an. REM deshalb, weil dieser Anweisung beliebige Zeichen folgen dürfen (REM-Kommentar) und Basic-Programme mit Maschinenprogrammen in REM-Kommentaren friedlich „zusammenleben" können.

Folgen einer REM-Anweisung z. B. 17 beliebige Zeichen (Buchstaben oder Zahlen), dann sind 17 Byte für ein Maschinenprogramm reserviert. Wie aber kommt das Programm in den REM-Kommentar? Das geht nur mit POKE-Befehlen. Die aber verlangen nach den Adressen der reservierten Speicherzellen.

Ist der REM-Kommentar in der ersten Programmzeile eines Basic-Programms untergebracht, dann ist die Sache einfach. Der Programmspeicher beginnt nämlich immer mit Adresse 16509. Zeilennummer, Daten über die Zeilenlänge und das Schlüsselwort REM nehmen dann insgesamt 5 Byte in Anspruch. Die Adresse des ersten Zeichens im REM-Kommentar hat damit den Wert 16514, und genau unter dieser Adresse darf das erste Byte eines Maschinenprogramms untergebracht werden.

Liegen die Bytes eines Maschinenprogramms als Dezimalwerte vor, so kann man die Werte einzeln oder besser mit einer FOR-NEXT-Schleife per POKE-Befehl in den REM-Kommentar unterbringen. Der Kommentar ist danach mit dem Maschinenprogramm „geladen", das z. B. mit RAND USR 16514 aufgerufen wird. Das Eingabe-Programm, das den POKE-Vorgang vorgenommen hat, darf jetzt wieder gelöscht werden.

Liegt ein Maschinenprogramm als Folge von Hex-Codes vor, dann ist zu deren Eingabe das Programm aus *Bild 1* einzutippen. Sollten die Zeilen nach dem

```
100 INPUT I$
105 IF I$="" THEN STOP
110 IF LEN I$<>5 THEN RUN
115 PRINT I$;" = ";
120 IF I$(5)="H" THEN GOTO 300
130 LET I=VAL I$
140 LET Z=I-256*INT (I/256)
150 GOSUB 280
160 LET I$=Z$
170 LET Z=INT (I/256)
180 GOSUB 280
190 LET I$=Z$+I$+"H"
200 PRINT I$
210 RUN
280 LET Z$=CHR$ (28+INT (Z/16))+CHR$ (Z-16*INT (Z/16)+28)
290 RETURN
300 LET Z$=I$(1 TO 2)
310 GOSUB 480
320 LET I=Z
330 LET Z$=I$(3 TO 4)
340 GOSUB 480
350 LET I=256*I+Z
360 PRINT I
370 RUN
480 LET Z=16*(CODE Z$(1)-28)+CODE Z$(2)-28
490 RETURN
```

② **Umrechnungsprogramm:** Es läßt sich auf Dezimal- und Hexadezimalzahlen anwenden

```
10 REM 0000000000000000000000000000000000000000000000000000000000000000
20 LET A$=""
30 LET ADR=16514
40 FOR N=0 TO LEN A$-2 STEP 2
50 POKE ADR+INT (N/2),16*(CODE A$(N+1)-28)+CODE A$(N+2)-28
60 NEXT N
70 STOP
80 SAVE "USR"
90 LIST
95 STOP
```

① **Hex-Code-Eingabe:** Im String A$ untergebrachte Hex-Codes werden in Dezimalzahlen umgerechnet und in die vom REM-Kommentar reservierten Speicherzellen geladen

REM-Kommentar (Zeile 10) schon von einem anderen Basic-Programm belegt sein, so können die Zeilen 20 bis 95 auch an dieses Programm angehängt werden.

Die einzugebenden Hex-Codes müssen nun in der Zeichenkette A$ (Zeile 20) untergebracht werden: z. B. LET A$="03EF3E03". Dieses 4 Byte lange Programm würde in Zeile 10 nur die Eingabe von vier Nullen verlangen. Jetzt darf das Hex-Code-Eingabeprogramm mit RUN 20 (bzw. entsprechende Zeilennummer) gestartet werden. Nach dem Abbruch mit der STOP-Meldung hat Zeile 10 ihr Aussehen verändert, enthält sich doch nun den Maschinencode. Damit hat das Eingabeprogramm seinen Zweck erfüllt, und es darf gelöscht werden. Die REM-Zeile aber muß selbstverständlich erhalten bleiben.

Das Programm in Bild 1 nimmt eine Hexadezimal-/Dezimal-Wandlung vor, denn die POKE-Befehle verlangen immer nach Dezimalzahlen. In *Bild 2* ist noch ein Programm gezeigt, das einem die Umrechnung zwischen beiden Zahlensystemen abnimmt. Es verlangt nach dem Start nach vierstelligen Hex-Zahlen, die mit einem H abgeschlossen werden müssen (z. B. 003Fh) oder nach fünfstelligen Dezimalzahlen. -ll

ZX-81-Softwaretip:

Listen am laufenden Band

Allen, die sich schon einmal an den Grenzen des LIST-Befehls beim ZX 81 gestoßen haben, zeigt das hier vorgestellte Programm einen Ausweg. Es ermöglicht ein kontinuierliches Auflisten – d. h. das Programmlisting läuft am Bildschirm ohne Abbruch so lange hoch, bis man es stoppt.

Zentraler Teil des LIST-Programms ist ein Maschinenprogramm, mit dem wir dem ZX 81 zu Leibe rücken. Platz dafür ist in Zeile 1 (*Bild*). Mit den Zeilen 2 bis 6 wird die Startnummer, ab der gelistet werden soll, eingegeben, für den Hausgebrauch des ZX 81 um 1 verringert und dann in der Systemvariablen E-PPC (Zeilennummer, in der der Cursor steht) abgespeichert.

Dann wird mit RAND USR 16514 das erste Maschinenprogramm aufgerufen. Es stellt die höchste im Programm auftretende Zeilennummer fest und lädt sie in die Speicherzellen 16507 und 16508, die zwar im Bereich der Systemvariablen liegen, aber vom Computer nicht belegt werden. Im Programm ist diese Information nötig, um nach Ausgabe der letzten Zeile das Programm zu unterbrechen.

Die Zeilen 8 bis 10 dienen dem Zweck, die PRINT-Position auf Zeile 18 zu verschieben. Dazu wird die Zahl der Leerzeilen im unteren Teil des Bildschirms durch Poken von DF-SZ (Zahl der Zeilen im unteren Teil des Bildschirms) von zwei auf fünf erhöht. Anschließend wirkt auf die Zeilen 0 bis 18 der SCROLL-Befehl. Jetzt wird DF-SZ wieder auf 2 gesetzt. Die PRINT-Position bleibt davon unberührt in Zeile 18. Es können also nun vier Zeilen im unteren Teil des Bildschirms ausgegeben werden.

In Zeile 11 wird überprüft, ob bereits die letzte Zeile ausgegeben wurde, oder ob durch Drücken von NEWLINE das Programm gestoppt werden soll. Wird kein Halt gefordert, ruft der ZX 81 das zweite Maschinenprogramm auf, das bei Adresse 16539 beginnt. Es verschiebt den Programm-Cursor zur folgenden Programmzeile und gibt sie auf dem Bildschirm aus. Die abschließenden SCROLL-Befehle sorgen dafür, daß auch längere Programmzeilen richtig ausgegeben werden. Nach einem Abbruch läßt sich das Auflisten durch Eingabe von CONT fortsetzen. Dabei werden allerdings einige Programmzeilen „verschluckt".

In den Zeilen 50 bis 90 ist der Maschinencodelader aus FUNKSCHAU 12/83 untergebracht. Um die Maschinencodezeile übersichtlicher zu machen, wurde er etwas abgeändert, so daß die Hex-Codes nun durch ein Leerzeichen (SPACE) getrennt einzugeben sind!

Zum Gebrauch: Erst das Programm eingeben und dann sicherheitshalber durch GOTO 17 abspeichern. Mit GOTO 50 sind anschließend die Maschinenprogramme zu laden. Nach dem Start durch RUN läßt sich die Startzeile eintippen (NEWLINE nur kurz drücken, sonst stoppt der Computer). Wenn jetzt kein Listing läuft, liegt ein Eingabefehler vor. Ist dagegen alles in Ordnung, können die Zeilen 30 bis 90 gelöscht und darf das Programm endgültig abgespeichert werden.

Im Listing fällt auf, daß jede Zeile mit Programm-Cursor ausgegeben wird, aber nach Stopp und erneutem Drücken von NEWLINE zeigt sich, daß er in der zuletzt ausgegebenen Zeile steht. Für Programmzeilen großer Länge muß statt des Wertes 5 in Zeile 8 ein höherer Wert gewählt werden, und wem es zu schnell geht, der muß eine Pause einfügen. Das fertige Programm sollte vor der Eingabe größerer Programme geladen werden. Da es am Anfang steht, genügt dann zum Aufrufen die Eingabe von RUN. Andreas Brecht

```
   1 REM REMZEILE ZUR AUFNAHME U
ON 2 MASCHINENCODEPROGRAMMEN
   2 PRINT "STARTZEILE EINGEBEN"
,"STOP MIT NEWLINE"
   3 INPUT START
   4 LET START=START-1
   5 POKE 16394,START-256*INT (S
TART/256)
   6 POKE 16395,INT (START/256)
   7 RAND USR 16514
   8 POKE 16418,5
   9 SCROLL
  10 POKE 16418,2
  11 IF CODE INKEY$=118 OR (PEEK
 16507<=PEEK 16394 AND PEEK 1650
8<=PEEK 16395) THEN STOP
  12 RAND USR 16539
  13 SCROLL
  14 SCROLL
  15 SCROLL
  16 GOTO 8
  17 SAVE "LIS■"
  30 LIST
  50 LET A$="01 04 04 2A 0C 40 2
B 2B 3E 77 CD ED 59 23 23 7E 32
7C 40 23 7E 32 7B 40 C9 2A 0A 40
 23 22 0A 40 CD DB 09 7E 22 0B 4
0 23 7E 28 32 0A 40 1E 00 CD 45
07 C9"
  60 LET ADR=16514
  70 FOR N=0 TO LEN A$-2 STEP 3
  80 POKE ADR+INT (N/3),16*(CODE
 A$(N+1)-28)+CODE A$(N+2)-28
  90 NEXT N
```

LIST-Programm: Damit kann man ein Programm vollständig (ohne Zwangsabbruch bei vollem Bildschirm) listen lassen. Hat das Eingabeprogramm ab Zeile 30 seinen Zweck erfüllt, darf es gelöscht werden

ZX-81-Betriebssystem:

RAMTOP-Story

In einen Teufelskreis gerät man, wenn es gilt, beim ZX 81 Programme vor dem Killerbefehl NEW zu schützen. Lesen Sie die RAMTOP-Story, um aus diesem Teufelskreis zu entkommen.

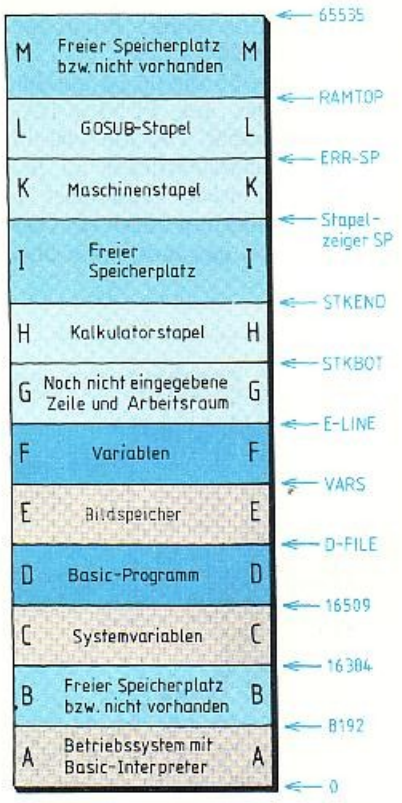
Anhand der Problemstellung, nach dem Laden eines Programms die RAMTOP-Adresse zu ändern, um dieses Programm vor dem Überschreiben bzw. dem Befehl NEW zu schützen, werden im folgenden einige Eigenschaften und Eigenarten des Betriebssystems beschrieben und Programmiertricks für das Arbeiten in Maschinensprache verraten.

RAMTOP-Ändern ohne NEW

Bei sehr vielen Maschinenprogrammen ist es auch erforderlich, bereits vor dem Laden die RAMTOP-Adresse zu ändern. Damit wird ein geschützter Bereich innerhalb des Speichers geschaffen, in dem das Programm vor Zerstörung sicher ist. Vergißt man diese Vorarbeit – und wem ist das noch nicht passiert – so geschehen gewöhnlich die merkwürdigsten Dinge. Bei manchen Programmen, die wohl den Netzteilstekker vor zu häufigem Ziehen bewahren wollen, erhält der vergeßliche ZX-81-Benutzer die freundliche Botschaft „RAMTOP NOT LOWERED" oder ähnliches.

Dann heißt es, die Cassette zurückzuspulen, versäumtes RAMTOP-Ändern nachzuholen – bloß nicht NEW vergessen – und erneut den LOAD-Befehl zu geben. Und während unser vergeßlicher Freund nun wieder einige Minuten auf den flimmernden Bildschirm starrt, träumt er von Programmen, die auf seine Vergeßlichkeit Rücksicht nehmen und sich selbst schützen, oder von einer kleinen Hilfsroutine, die man vielleicht in alle vorhandenen Programme einbauen kann und die dasselbe bewirkt. Aber wie soll das nur funktionieren, wenn man doch immer den NEW-Befehl geben muß?

Es geht! – Um zu verstehen, wie, schauen wir uns zunächst einmal in *Bild 1* die Gliederung des Speichers an, wobei von einem Vollausbau auf 64 KByte ausgegangen wird. Ist ein kleineres oder gar kein Modul angeschlossen, so ist der mit B bezeichnete Speicherbereich in Bild 1 nicht vorhanden. Entsprechendes gilt für den Bereich M, wenn der zur Verfügung stehende Speicher kleiner als 16 KByte ist.



① **Speicherverteilung des ZX 81:** Der Beitrag zeigt detailliert, wie sich der Computer nach dem Einschalten den verfügbaren Speicher selbst einteilt

Für den eiligen Leser

Wer auf dem ZX 81 Maschinenprogramme laufen läßt, kennt das Problem genau: Zum Sichern der Programme genügt es nicht, RAMTOP tiefer zu setzen, sondern es muß zusätzlich NEW eingetippt werden. Der Beitrag zeigt, warum das so ist. NEW fegt jedoch den Programmspeicher leer und löscht z. B. gerade das Programm, das man eigentlich in den Bereich oberhalb von RAMTOP kopieren wollte. Aus diesem Grund ist RAMTOP unmittelbar nach dem Einschalten des ZX 81 tiefer zu setzen; erst danach darf man das zu kopierende Programm laden. Aber wer hat diese Reihenfolge nicht schon mißachtet?!

Lädt man das im Beitrag beschriebene Hilfsprogramm in den ZX 81 (ideal wäre die Unterbringung in dem auf Seite 48 beschriebenen Zusatzspeicher), dann steht quasi ein neuer Befehl zum Ändern von RAMTOP ohne NEW bereit. Eine Mini-Version des Programms läßt sich in fertige Maschinenprogramme einbauen, so daß sich diese selber schützen können.

Der Bereich A ist vom Festwertspeicher (ROM) belegt, der das Betriebssystem und den Basic-Interpreter enthält. Dies ist der einzige Speicherabschnitt, den wir nicht durch POKE verändern können.

Ein „seltsames" Programm

Wird der ZX 81 eingeschaltet, so arbeitet der Mikroprozessor die im ROM befindlichen Befehle ab der Adresse 0 ab. Dabei führt er zunächst einen Speichertest durch. Er schreibt ab der Adresse 32767 rückwärts bis zur Adresse 16384 in alle Speicherzellen eine 2. Was passiert aber, wenn es die höheren Speicherzellen gar nicht gibt, weil wir vielleicht kein Erweiterungsmodul angeschlossen haben?

Wie schon öfter in der Fachliteratur erwähnt, wird beim ZX 81 der Adreßbus nicht vollständig decodiert. Das hat zur Folge, daß auch dann etwas passiert,

wenn wir Speicherzellen ansprechen, die nicht vorhanden sind. Ist z. B. der Abschnitt B in Bild 1 nicht vorhanden, so ergibt (in der Grundversion des ZX 81) das Abfragen der Speicherzellen dieses Bereiches den Inhalt des ROMs.

In der Grundversion des ZX 81 (1 KByte RAM) hat die höchste vorhandene Speicherzelle die Adresse 17407. Um festzustellen, was passiert, wenn wir höhere Adressen ansprechen, geben wir als erste Programmzeile ein:

1 REM XXXXXXX

Nun führen wir von Hand die Befehle

POKE 16514,56
POKE 17539,42
POKE 18564,49
POKE 19589,57
POKE 20614,56
POKE 21639,38
POKE 22664,50

aus und schauen anschließend die erste Programmzeile an: Das ist wirklich seltsam und eine besondere Eigenart des ZX 81.

Haben wir nur 1 KByte zur Verfügung, also den Bereich zwischen den Adressen 16384 und 17407, so erreichen wir diesen Abschnitt auch über Adressen zwischen 17408 und 18431, aber auch über Adressen zwischen 18432 und 19455 usw., also immer in Schritten von 1024 Byte (1 KByte).

Haben wir die 16-KByte-Speichererweiterung angeschlossen und können damit den Bereich zwischen den Adressen 16384 und 32767 benutzen, so erreichen wir auch diesen Abschnitt noch zweimal in 16-KByte-Schritten im Adreßbereich 32768 bis 49151 und 49152 bis 65535.

So teilt sich der ZX 81 den Speicher ein

Doch nun zurück zum Speichertest, den der Mikroprozessor nach dem Einschalten durchführt. Existieren die höheren Speicherzellen nicht, so nimmt er, wie wir eben gesehen haben, die entsprechenden Zellen, die vorhanden sind.

Anschließend macht der Mikroprozessor ab Adresse 16384 aufwärts bis zur Adresse 32767 folgendes: Er verringert den Inhalt der anstehenden Speicherzelle zweimal um 1 und überprüft, ob jetzt eine 0 in der Zelle steht. Wenn dem so ist, nimmt er die nächste Speicherzelle

```
ADR. MNEMONIC          HEX-CODES

4082 LD A,255          3EFF          NMI-Generator ausschalten.
4084 OUT 253,A         D3FD          CDFLAG für NMI-Generator-
4086 LD (IY+059),064   FD363B40      Einschaltung vorbereiten.
408A LD (16444),SP     ED733C40      Stapelzeiger abspeichern.

408E CALL 00047        CD2F00        Lage des Programms im
4091 LD HL,(16444)     2A3C40        Speicher feststellen.
4094 DEC HL            2B
4095 LD D,(HL)         56
4096 DEC HL            2B
4097 LD E,(HL)         5E
4098 LD HL,00177       21B100        Lage des Zeigeranpassungs-
409B ADD HL,DE         19            programms berechnen und
409C LD (16446),HL     223E40        abspeichern.

409F LD DE,00008       110800        Zeigeranpassungsprogramm
40A2 LD (HL),000       3600          auf „RAMTOP erhöhen"
40A4 ADD HL,DE         19            einstellen.
40A5 LD (HL),025       3619
40A7 ADD HL,DE         19
40A8 LD (HL),025       3619
40AA LD HL,(16386)     2A0240        Feststellen, ob RAMTOP
40AD INC HL            23            von Hand geändert wurde.
40AE INC HL            23
40AF INC HL            23
40B0 LD A,(HL)         7E
40B1 CP 062            FE3E
40B3 JR NZ,$40AE       20F9
40B5 INC HL            23
40B6 AND A             A7
40B7 LD DE,(16388)     ED5B0440
40BB SBC HL,DE         ED52
40BD CALL NZ,000B6     C45600        Wenn ja, Abbruch mit
40C0 DFB 20            2000          Fehlermeldung X.

40C2 LD HL,(16507)     2A7B40        Wird neue RAMTOP-Adresse
40C5 LD A,H            7C            „Spar"-Bildschirm
40C6 CP 077            FE4D          bewirken?
40C8 JR C,$40D8        38 0E         Wenn ja, Sprung nach 40D8.
40CA LD A,D            7A            Besteht momentan
40CB CP 077            FE4D          „großer" Bildschirm?
40CD JR NC,$40E6       3017          Wenn ja, Sprung nach 40E6.
40CF ADD A,002         C602          Darf Bildschirm
40D1 CP H              BC            vergrößert werden?
40D2 JR C,$40E6        3812          Wenn ja, Sprung nach 40E6.
40D4 CALL 000B6        CD5600        Abbruch mit
40D7 DFB 21            21            Fehlermeldung Y.

40D8 LD (16388),HL     220440        Falls erforderlich, auf
40DB PUSH HL           E5            „Spar"-Bildschirm
40DC PUSH DE           D5            umschalten.
40DD CALL 02602        CD2A0A
40E0 POP DE            D1
40E1 POP HL            E1
40E2 LD (16388),DE     ED530440
40E6 AND A             A7            Neue RAMTOP-Adresse
40E7 SBC HL,DE         ED52          mit alter vergleichen.
40E9 JR Z,$4158        286D          Wenn gleich, Sprung nach 4158.
40EB LD BC,(16444)     ED4B3C40      Wenn neue Adresse kleiner
40EF JR C,$40FF        38 0E         als alte, Sprung nach 40FF.
40F1 EX DE,HL          EB
40F2 SBC HL,BC         ED42          Feststellen, ob Kopie
40F4 SBC HL,DE         ED52          durchgeführt werden darf.

40F6 JR C,$4126        382E          Wenn ja, Sprung nach 4126.
40FB CALL 02602        CD2A0A        Bildschirm auf alte Größe
40FB CALL 00086        CD5600        setzen und Abbruch mit
40FE DFB 22            22            Fehlermeldung Z.
```

```
40FF ADD HL,DE          19          Feststellen, ob Kopie
4100 AND A              A7          durchgeführt werden darf.
4101 SBC HL,BC          ED42
4103 JR NC,$40F8        30F3
4105 EX DE,HL           EB
4106 AND A              A7
4107 SBC HL,BC          ED42
4109 SBC HL,DE          ED52
410B LD DE,(16412)      ED5B1C40
410F ADD HL,DE          19
4110 LD DE,00072        114B00
4113 ADD HL,DE          19
4114 SBC HL,BC          ED42
4116 JR NC,$40F8        30E0        Wenn nein, Sprung nach 40F8.

411B LD HL,(16446)      2A3E40      Zeigeranpassungsprogramm
411B LD DE,0000B        110B00      auf „RAMTOP erniedrigen"
411E LD (HL),235        36EB        einstellen.
4120 ADD HL,DE          19
4121 LD (HL),237        36ED
4123 ADD HL,DE          19
4124 LD (HL),237        36ED
4126 LD HL,(16388)      2A0440      Kopie durchführen.
4129 PUSH HL            E5
412A AND A              A7
412B SBC HL,BC          ED42
412D LD B,H             44
412E LD C,L             4D
412F POP HL             E1
4130 LD DE,(16507)      ED5B7B40
4134 DEC HL             2B
4135 DEC DE             1B
4136 LDDR               EDB8
4138 LD HL,(16507)      2A7B40      Zeiger anpassen.
413B LD DE,(16388)      ED5B0440
413F LD (16388),HL      220440
4142 NOP                00
4143 AND A              A7
4144 SBC HL,DE          ED52
4146 EX DE,HL           EB
4147 LD HL,(16386)      2A0240
414A NOP                00
414B LD D,D             52
414C LD (16386),HL      220240
414F LD HL,(16444)      2A3C40
4152 NOP                00
4153 LD D,D             52
4154 LD SP,HL           F9
4155 CALL 02602         CD2A0A      Bildschirm auf neue Größe einstellen,
415B CALL 00519         CD0702      NMI-Generator einschalten
415B RET                C9          und Rücksprung zu Basic.
```

② **Programm zum Ändern von RAMTOP ohne NEW:** Diese Version kann RAMTOP nach oben/unten verschieben, wobei es unzulässige Werte nicht abweist

usw. Trifft er auf die erste nicht vorhandene Zelle, so nimmt er, wie wir wissen, die entsprechende vorhandene in einem unteren Speicherbereich. Da er diese aber bereits vorher auf 0 gesetzt hat, führt das zweimalige Verringern des Inhalts dieser Speicherzelle auf einen von Null verschiedenen Wert.

Dies erkennt der Mikroprozessor und bricht den Speichertest ab. Die Adresse der ersten nicht vorhandenen Speicherzelle ist also gefunden. Diese Adresse wird im Bereich C (Bild 1) in der Systemvariablen RAMTOP abgelegt.

Haben wir mindestens die 16-KByte-Erweiterung angeschlossen, so werden natürlich alle Speicherzellen gefunden, da ja bis zur Adresse 32767 gesucht wird und bis dahin alle Zellen existieren. In diesem Fall bekommt RAMTOP den Wert 32768. Diesen Wert bekommt RAMTOP aber auch dann, wenn eine 32-KByte-Erweiterung oder eine noch größere angeschlossen ist. Auf diese Weise entsteht der freie Speicherplatz im Bereich M in Bild 1.

Nach dem Speichertest erhält die letzte vorhandene Speicherzelle bzw. die Zelle 32767 den Wert 62, was einem Y entspricht und als Endmarkierung dient. Der GOSUB-Stapel (Bereich L) enthält also nur diesen Wert und der Maschinenstapel (Bereich K) bleibt noch leer. Der Systemvariablen ERR-SP und dem Stapelzeiger SP werden die Anfangsadressen der entsprechenden Bereiche zugeteilt. Werden diese Abschnitte später beim Abarbeiten von Programmen gefüllt, so dehnen sie sich nach unten aus, ERR-SP und SP bekommen neue Werte zugeteilt, und der freie Speicherplatz im Bereich I wird kleiner.

Nachdem nun am oberen Ende des Speichers Ordnung herrscht, kümmert sich das Betriebssystem um den unteren Teil. Es legt zunächst den Beginn des Bildspeichers fest und schreibt die Anfangsadresse des Bereiches E in die Systemvariable D-FILE. Da noch kein Basic-Programm eingegeben wurde, der Bereich D also noch leer ist, bekommt D-FILE den Wert 16509.

Die Sache mit dem Bildspeicher

Anschließend wird ein „Spar"-Bildschirm aufgebaut, indem der Bildspeicher-Bereich E mit 25 NEWLINE-Codes gefüllt wird. Nun ist auch die Anfangsadresse des Variablen-Bereiches F bekannt und kann in der Systemvariablen VARS abgelegt werden. In den Variablen-Bereich wird ein einziges Zeichen (negativ Space) gebracht, das wieder als Endmarkierung dient. Die nächste Adresse wird als Anfangsadresse des Arbeitsraumes (Bereich G) der Systemvariablen E-LINE zugeteilt.

In den Arbeitsraum wird der Cursor und ein NEWLINE-Code gesetzt. Die darauffolgende Adresse wird sowohl der Systemvariablen STKBOT als auch der Variablen STKEND zugeteilt. Der Kalkulatorstapel (Bereich H) bleibt also noch leer. Zu diesem Zeitpunkt hat der freie Speicherplatz im Bereich I also seine maximale Größe.

Dies kann sich jedoch sofort ändern: Bevor sich das System mit dem gewohnten K-Cursor bereitmeldet, führt es nämlich noch die CLS-Routine (Bildschirm löschen) aus. Diese Routine schaut nach, ob die Systemvariable RAMTOP einen verfügbaren Speicher von mindestens 4 KByte angibt. Ist das der Fall, so erweitert sie den „Spar"-Bildschirm auf volle Größe, indem sie im Bildspeicher-Bereich E zwischen je zwei NEWLINE-Codes 32 Leerzeichen setzt. Hierbei werden die Bereiche F, G und H nach oben verschoben, die Systemvariablen VARS, E-LINE, STKBOT und STKEND erhalten

entsprechend geänderte Werte, und der freie Speicherplatz im Bereich I wird kleiner.

Es ist für spätere Betrachtungen wichtig, zu wissen, daß die CLS-Routine auch bei jedem von Hand eingegebenen Kommando vor der Bearbeitung dieses Kommandos automatisch ausgeführt wird und daß dabei entsprechend dem Wert von RAMTOP entweder ein aus nur 25 NEWLINE-Codes bestehender „Spar"-Bildschirm oder ein 792 Byte beanspruchender „Groß"-Bildschirm aufgebaut wird. Das ZX-81-Handbuch erwähnt zwar die Möglichkeit, zwischen „Spar"- und „Groß"-Bildschirm umzuschalten, verschweigt jedoch die Nachteile und Gefahren. Diese werden später noch ausführlich beschrieben.

Achtung Interrupt

Bis auf eine Ausnahme kennen wir nun alle zur Lösung der uns gestellten Aufgabe notwendigen Eigenschaften des Betriebssystems. Nicht erwähnt wurde bisher, daß der Mikroprozessor sofort nach dem Einschalten, also noch vor dem Speichertest, einen ganz speziellen Befehl ausführt und diesen erst direkt vor der Ausführung der CLS-Routine wieder rückgängig macht, wonach er sich mit dem K-Cursor bereitmeldet.

Dieser spezielle Befehl schaltet den in einem zusätzlichen Chip befindlichen NMI-Generator aus, um zu verhindern, daß von dort nicht maskierbare Interrupts an den Mikroprozessor geschickt werden. Obwohl sich das furchtbar kompliziert anhört, werden wir gleich wissen, was damit gemeint ist.

Ein Interrupt ist ein Unterbrechungssignal, das über eine eigene Leitung vom Mikroprozessor empfangen werden kann. Erhält der Mikroprozessor ein solches Signal, so führt er den gerade anstehenden Befehl noch aus, legt die Adresse des nächsten Befehls auf den Maschinenstapel und verzweigt zu einer vorbestimmten Adresse. Hier befindet sich eine Routine, nach deren Abarbeitung der Mikroprozessor wieder zu der auf dem Stapel abgelegten Adresse zurückkehrt und seine unterbrochene Arbeit fortsetzt.

Welche Befehlsfolge ist nun aber innerhalb unseres Computers von so großer Wichtigkeit, daß alle anderen Arbeiten unterbrochen werden müssen? Es ist die Fernsehbildausgabe! Da der ZX 81 keinen speziellen Video-Prozessor hat, muß der Mikroprozessor diese Arbeit mitübernehmen.

Die entsprechenden Befehlsroutinen sind im Betriebssystem enthalten. Um eine einwandfreie Fernsehbild-Ausgabe zu erhalten, müssen diese Routinen in regelmäßigen Abständen 50mal in der Sekunde abgearbeitet werden. Deshalb schickt der NMI-Generator jede fünfzigstel Sekunde einen Interrupt an den Mikroprozessor.

Und warum wird der NMI-Generator beim Urstart zunächst ausgeschaltet? Das läßt sich sehr leicht einsehen, wenn man berücksichtigt, daß die Video-Routinen unter anderem wissen müssen, wo der Bildspeicher beginnt. Außerdem müssen sie im Bildspeicher NEWLINE-Codes finden, um erkennen zu können, wo eine Zeile zu Ende ist. Dies alles ist aber zu Beginn noch nicht festgelegt worden. Ein Abarbeiten der Interrupt-Routine zu diesem Zeitpunkt würde also zwangsläufig zum Absturz des Systems führen.

Jetzt gerät RAMTOP in Bewegung

Nachdem wir nun einen Einblick in das Betriebssystem des ZX 81 gewonnen und uns mit der Speicherorganisation dieses Computers vertraut gemacht haben, wollen wir uns einmal überlegen, was passiert, wenn wir der Systemvariablen RAMTOP durch POKE einen anderen Wert zuweisen, aber vergessen, NEW zu geben.

Schauen wir uns dazu noch einmal Bild 1 an. Da die Speicherbereiche A bis L danach weiterhin die gleichen Inhalte haben und auch die dazugehörigen Systemvariablen noch immer auf dieselben Adressen zeigen, wird in der Regel gar nichts passieren. Ändern wir allerdings (in der Grundversion) den RAMTOP-Inhalt von einem Wert unter 4 KByte auf einen Wert darüber, so wird, wie erwähnt, der Bildspeicher auf volle Größe erweitert.

Für das Hochschieben der Bereiche F, G und H könnte nun im Bereich I zu wenig Platz vorhanden sein, falls ein längeres Programm eingegeben wurde oder sehr viele Variablen vereinbart wurden. Trotzdem führt das Betriebssystem die Vergrößerung des Bildschirmspeichers durch. Danach befolgt es Befehle nur noch auf sehr exotische Art oder verabschiedet sich gleich. Manchmal bringt es auch die verspätete Fehlermeldung 4, als wenn es uns sagen wollte, warum es aussteigt.

Aber selbst wenn wir dafür Sorge tragen, daß im Bereich I genug Platz für die Bildschirmvergrößerung vorhanden ist, so passieren doch recht unbefriedigende Dinge, die das Handbuch verschweigt. Wer mindestens 16 KByte zur Verfügung hat, gebe einmal folgendes Programm ein:

```
10 DIM A (3000)
20 POKE 16389,76
30 CLS
40 LIST
50 POKE 16389,128
60 CLS
70 LIST
```

Nach dem Start mit RUN darf man ruhig einen Spaziergang machen. Das System wird sich nämlich erst nach geraumer Zeit mit der Auskunft 0/70 zurückmelden. Der Grund für die „enorme" Geschwindigkeit beim Ausführen der Zeile 60 ist ebenfalls im Betriebssystem zu suchen.

Wenn das System den Bildspeicher vergrößert, so verschiebt es Zeichen für Zeichen die Bereiche F, G und H nach oben. Für jedes Leerzeichen, das in den Bildspeicher gebracht wird, werden in unserem Fall 3000 Zahlen, die wiederum jeweils aus 5 Byte bestehen, verschoben. Es werden somit über 750mal rund 15 000 Zeichen bewegt. Und das dauert eben seine Zeit.

Außer diesen unangenehmen Dingen wird aber nichts passieren, wenn wir den Inhalt der Systemvariablen RAMTOP durch POKE verändern. Das läßt sich auch sehr leicht einsehen. Nehmen wir einmal an, wir hätten das 16-KByte-Speichermodul angeschlossen. RAMTOP erhält dann nach dem Einschalten den Wert 32768. Nun wollen wir für ein 768 Byte langes Maschinenprogramm einen geschützten Raum am Ende des Speichers schaffen. Dazu geben wir RAMTOP den Wert 32000, indem wir den Befehl

```
POKE 16389,125
```

ausführen. RAMTOP zeigt nun auf eine Speicherzelle, die wohl im Bereich I (Bild 1) liegen wird. Da jetzt aber die Systemvariable ERR-SP und der Maschinenstapelzeiger SP immer noch die ursprünglichen Werte haben, befinden sich der GOSUB-Stapel und der Maschinenstapel nach wie vor an derselben Stelle.

Wir haben also unser Ziel, einen geschützten Bereich zu schaffen, nicht er-

reicht. Wenn wir jetzt nämlich in den angeblich geschützten Bereich unser Maschinenprogramm hineinschreiben würden, so veränderten wir auch den Inhalt des Maschinenstapels, was zwangsläufig zum Absturz des Systems führen muß. Dies können wir anhand eines kleinen Programms testen, das in die Speicherzellen 32000 bis 32767 den Wert 222 schreibt:

```
10 POKE 16389,125
20 FOR N=32000 TO 32767
30 POKE N,222
40 NEXT N
```

Hätten wir nun nach dem Tiefersetzen von RAMTOP den NEW-Befehl gegeben, so wäre für uns ein Bereich reserviert worden, ohne daß die Gefahr des Systemabsturzes bestanden hätte. Woran liegt das?

Wird der NEW-Befehl gegeben, so macht der Mikroprozessor alles, was er auch nach dem Einschalten macht, allerdings mit einem kleinen Unterschied: Wenn er beim Speichertest die erste nicht vorhandene Speicherzelle sucht, hört er mit der Suche nicht bei der Adresse 32767 auf, sondern bei der Adresse, auf die RAMTOP zeigt. Und die haben wir ja gerade vorher nach unseren Wünschen festgelegt. Dadurch werden die Bereiche L und K dorthin gelegt, wo sie hingehören, und die freien Speicherbereiche I und M sind so groß oder so klein, wie wir es wünschen.

So läßt sich NEW vermeiden

Wie können wir nun nach dem Laden eines Programms das gleiche Ziel erreichen, ohne den für das geladene Programm tödlichen NEW-Befehl geben zu müssen?

Wir verschieben die Speicherbereiche L und K nach unten, indem wir sie in den Bereich I kopieren. Anschließend müssen wir der Systemvariablen ERR-SP und dem Maschinenstapelzeiger SP die neuen Anfangsadressen der kopierten Bereiche zuteilen. Das ist alles. Leider können wir diese leichte Aufgabe nicht durch ein Basic-Programm lösen. Der Maschinenstapelzeiger SP ist nämlich keine Systemvariable, sondern ein internes Register des Mikroprozessors. Und dieses Register können wir nur in Maschinensprache ändern.

Das Maschinenprogramm in *Bild 2* erledigt die gestellte Aufgabe in Bruchteilen einer Sekunde, ermöglicht aber darüber hinaus nicht nur ein Verringern von RAMTOP, sondern auch ein beliebiges Erhöhen, soweit dadurch nicht andere Speicherbereiche zerstört werden könnten. Eine notwendig werdende Bildschirmvergrößerung führt es in kürzester Zeit durch. Es läßt sich sehr leicht bedienen, gibt eigene Fehlermeldungen aus und kann in jedem beliebigen Speicherbereich ohne irgendeine Änderung betrieben werden. Es eignet sich daher als Erweiterung des vorhandenen Befehlssatzes.

Für diejenigen, die nur die lästige und leicht zu vergessende RAMTOP-Änderung vor dem Laden eines Maschinenprogramms umgehen möchten, wird später noch eine Miniversion dieses Programms beschrieben. Was bei der Miniversion jedoch alles vom Benutzer zu beachten ist, wird am deutlichsten bei der Besprechung der Maxiversion dieses Programms, die dem Anwender jede zusätzliche Überlegung abnimmt, ihn vor Fehlern warnt und fehlerhafte RAMTOP-Änderungen nicht ausführt.

Gleich zu Anfang ein Programmiertrick

Ehe wir das Programm im einzelnen durchleuchten, schauen wir uns einmal den letzten Teil an. In den Adressen 4142h, 414Ah und 4152h befindet sich der Befehl nop. Wir sind hier im Zeigeranpassungsprogramm, das der Systemvariablen ERR-SP und dem Stapelzeiger SP die neuen Werte zuweist. Je nachdem, ob RAMTOP erhöht oder verringert wird, müssen dort Additionen bzw. Subtraktionen durchgeführt werden.

Das Programm selbst ersetzt dann die nops durch Additions- bzw. Subtraktionsbefehle. Dazu muß das Programm aber wissen, an welche Adressen diese Befehle zu setzen sind. Das Programm muß also in der Lage sein, zu erkennen, in welchem Speicherbereich es sich befindet. Dies ist sicher ein interessanter Programmiertrick, mit dessen Erklärung die ausführliche Programmbeschreibung begonnen wird.

Das Register pc des Mikroprozessors, der Befehlszähler, der die Adresse des gerade auszuführenden Befehls enthält, ist leider nicht abrufbar. Folglich müssen wir einen anderen Weg wählen, um seinen Inhalt zu erfahren.

Wird ein Unterprogramm aufgerufen, z. B. durch einen *call*-Befehl, so wird bekanntlich die Adresse des nächsten Befehls auf den Stapel gelegt und anschließend zum Unterprogramm gesprungen. Wird dort ein ret-Befehl gefunden, wird die Adresse wieder vom Stapel geholt und die Befehlsausführung dort fortgesetzt.

Nun müssen wir noch wissen, daß etwas, was vom Stapel geholt wird, dort nicht automatisch gelöscht wird. Der Stapelzeiger zeigt zwar sofort auf den nächsten Stapelinhalt, aber der „oberste" Stapelinhalt wird erst in dem Moment gelöscht, wo wieder etwas auf den Stapel kommt. Es drängt sich nun folgende Idee auf:

Wir rufen durch einen *call*-Befehl eine Adresse im Betriebssystem auf, an der ein ret-Befehl steht. Dadurch wird die Adresse des auf *call* folgenden Befehls auf den Stapel gelegt. Danach holen wir uns sofort den Inhalt des Stapelzeigers in ein geeignetes Register und verringern dessen Inhalt um 2. Dieses Register zeigt nun auf eine Adresse, die mit der Adresse des auf *call* folgenden Befehls identisch ist. Auch diese Adresse können wir in ein beliebiges Register laden und dort weiterverwenden.

Somit weiß unser Programm, wo es sich selbst befindet. Wir müssen bei dieser Methode allerdings darauf achten, daß zwischendurch garantiert nichts auf den Stapel gelegt wird; denn das würde die Adresse, die uns interessiert, überschreiben.

Aber was ist mit den Video-Routinen, für die ja bekanntlich unser Programm ständig unterbrochen wird? Allein schon der Aufruf dieser Routinen durch einen Interrupt würde uns einen Strich durch die Rechnung machen, da ja mindestens die Rücksprungadresse auf den Stapel gelegt würde und somit das, was uns interessiert, zerstört wäre. Also müssen wir den NMI-Generator ausschalten, um Interrupts zu verhindern.

Genau das wird in den ersten zwei Zeilen des Programms in Bild 2 gemacht. Nehmen wir nun einmal an, wir wollten RAMTOP den Wert 30000 geben. Nach der herkömmlichen Art müßten wir dazu die folgenden 3 Befehle geben:

```
POKE 16388,48
POKE 16389,117
NEW
```

Wollen wir unser Programm benutzen, so geben wir statt dessen die Befehle:

```
POKE 16507,48
POKE 16508,117
RAND USR 16514
```

Betreiben wir unser Programm in einem anderen Speicherbereich, so steht hinter RAND USR natürlich eine andere Startadresse. Der neue RAMTOP-Wert steht nun in einem vom Betriebssystem nicht verwendeten Teil des Systemvariablen-Bereiches, und durch den RAND-USR-Befehl wird unser Programm gestartet.

Es schaltet zuerst den NMI-Generator aus und gibt der Systemvariablen CDFLAG einen Wert, der die spätere Wiedereinschaltung des NMI-Generators durch Aufruf einer Betriebssystem-Routine ermöglicht. Anschließend wird der Inhalt des Stapelzeigers in den Speicherzellen 16444/45 abgespeichert.

Die Speicherzellen 16444 bis 16475 werden vom Betriebssystem als Druckerpuffer verwendet und eignen sich daher normalerweise nicht zum Ablegen von Daten, da diese vom System sofort wieder zerstört würden. Da nach dem Abschalten des NMI-Generators der Mikroprozessor jedoch ausschließlich unsere Befehle abarbeitet, können wir diesen Bereich zur Ablage von Daten verwenden.

Durch den *call*-Befehl an der Adresse 408Eh wird nun ein ret-Befehl innerhalb des Betriebssystems aufgerufen, was bewirkt, daß die Adresse 4091h auf den Stapel gelegt wird. Der Programmteil 4091h bis 409Ch holt nun diese Adresse in der geschilderten Art und Weise vom Stapel und addiert dazu einen festen Wert. Der berechnete Wert ist die Adresse des ersten *nops* im Zeigeranpassungsprogramm und wird in den Speicherzellen 16446/47 abgespeichert. Unser Programm weiß nun, wo es sich befindet.

Der nächste Programmteil ersetzt die *nops* im Zeigeranpassungsprogramm durch Befehle, die dort eine Addition ermöglichen. Zu diesem Zeitpunkt geht das Programm also noch davon aus, daß RAMTOP erhöht werden soll.

Im folgenden Abschnitt schaut das Programm nach, ob RAMTOP vielleicht schon vorher von Hand geändert wurde, z. B. um einen „Spar"-Bildschirm zu bekommen. Das geschieht folgendermaßen: Es holt sich die Anfangsadresse des GOSUB-Stapels aus der Systemvariablen ERR-SP und sucht diesen nach der Endmarkierung, dem Y, ab. Die Adresse hinter dieser Endmarkierung müßte mit dem Wert von RAMTOP übereinstimmen.

Ist dies nicht der Fall, so wurde RAMTOP von Hand geändert, und unser Programm bricht seine Arbeit mit der Fehlermeldung X ab, indem mit *call* eine Routine im Betriebssystem aufgerufen wird, die folgendes bewirkt: Die Rücksprungadresse, in unserem Fall die Adresse 40C0h, wird vom Stapel geholt, und der Wert, der an dieser Adresse steht, hier 20h, wird als Fehlernummer interpretiert. Danach wird der NMI-Generator eingeschaltet und die entsprechende Fehlermeldung ausgegeben.

```
ADR. MNEMONIC          HEX-CODES

4082 LD (16507),SP     ED737B40
4086 LD DE,(16507)     ED5B7B40
408A LD HL,(16388)     2A0440
408D PUSH HL           E5
408E AND A             A7
408F SBC HL,DE         ED52
4091 LD B,H            44
4092 LD C,L            4D
4093 POP HL            E1
4094 LD DE,29696       110074
4097 DEC HL            2B
4098 DEC DE            1B
4099 LDDR              EDB8
409B LD HL,(16388)     2A0440
409E LD DE,29696       110074
40A1 LD (16388),DE     ED530440
40A5 AND A             A7
40A6 SBC HL,DE         ED52
40A8 EX DE,HL          EB
40A9 LD HL,(16386)     2A0240
40AC SBC HL,DE         ED52
40AE LD (16386),HL     220240
40B1 LD HL,(16507)     2A7B40
40B4 SBC HL,DE         ED52
40B6 LD SP,HL          F9
40B7 RET               C9
```

③ **Sichern von Maschinenprogrammen:** Diese Routine läßt sich in Maschinenprogramme einbauen, um diese oberhalb von RAMTOP unterzubringen

Obwohl wir einen *call*-Befehl gegeben haben, wird nicht mehr zu unserem Programm zurückgekehrt. Falls wir unser Maschinenprogramm von einem Basic-Programm aus aufgerufen haben, so wird auch dieses nicht mehr fortgesetzt. Die Fehlerbehandlung unseres Programms ist also identisch mit der des Basic-Interpreters.

Wurde RAMTOP nicht von Hand geändert, so wird der *call*-Befehl nicht ausgeführt, und unser Programm arbeitet an der Adresse 40C0h weiter. Jetzt versteht es den Wert 20h dort natürlich als Sprungbefehl. Deshalb steht dahinter noch der Wert 00. Beide Werte zusammen ergeben den Befehl *jr nz,40C2*. Unser Programm arbeitet also an der Adresse 40C2h weiter.

Hier überprüft es, ob der neue RAMTOP-Wert einen „Spar"-Bildschirm bewirken würde. Wenn dem so ist, springt es zum nächsten Programmteil, wo durch den Aufruf der CLS-Routine ein „Spar"-Bildschirm aufgebaut wird. Es ist wichtig, daß dies schon zu diesem Zeitpunkt geschieht. Überprüft nämlich unser Programm später, ob im Bereich I (Bild 1) genügend Platz für die Kopie des GOSUB- und des Maschinenstapels vorhanden ist, so könnte es feststellen, daß zu wenig Speicherplatz existiert, nur weil noch kein „Spar"-Bildschirm erzeugt wurde.

Wenn die neue RAMTOP-Adresse einen „Groß"-Bildschirm verlangt, überprüft unser Programm, ob zur Zeit schon ein solcher vorhanden ist, und springt in diesem Fall zum Programmteil mit der Adresse 40E6h. Besteht noch ein „Spar"-Bildschirm, wird festgestellt, ob im Bereich I genug Platz für eine Bildschirmvergrößerung vorhanden ist. Ist dies der Fall, wird ebenfalls an der Adresse 40E6h weitergearbeitet. Wenn nicht, bricht das Programm, wie zuvor beschrieben, die Abarbeitung mit der Fehlermeldung Y ab.

Im Programmteil ab der Adresse 40E6h wird der neue RAMTOP-Wert mit dem alten verglichen. Sind beide gleich, verzweigt das Programm zur Adresse 4158h, wo der NMI-Generator wieder eingeschaltet und ins Basic zurückgekehrt wird. Für die beiden anderen Fälle, die Erhöhung bzw. die Verringerung von RAMTOP, werden zwei verschiedene Programmteile durchlaufen und anschließend wird bei der Adresse 4126h weitergearbeitet.

Betrachten wir zunächst den Fall der RAMTOP-Erhöhung: An der Adresse 40E7h wurde bereits die Differenz zwischen neuem und altem Wert berechnet. Nun wird in Zeile 40F2h noch die Differenz zwischen dem alten RAMTOP-Wert und dem Inhalt des Stapelzeigers, also die Länge von GOSUB- und Maschinenstapel, ermittelt. Durch Subtraktion dieser beiden Differenzen stellt unser Programm fest, ob die Kopie in den Bereich M durchgeführt werden darf. Ist dies nicht möglich, wird mit der Fehlermeldung Z abgebrochen.

Schauen wir uns nun den Fall der RAMTOP-Verringerung an: Hier wird zuerst der neue RAMTOP-Wert mit dem Inhalt des Stapelzeigers verglichen. Wenn der neue RAMTOP-Wert auf eine höhere Adresse zeigt als der Stapelzeiger, wenn also beim Kopieren der Maschinenstapel zerstört werden könnte, bricht das Programm ebenfalls mit der Fehlermeldung Z ab.

Danach wird ab der Adresse 4107h festgestellt, ob die Kopie in den Bereich I

den Kalkulatorstapel überschreiben würde. Auch in diesem Fall wird die Fehlermeldung Z ausgegeben. Wenn die Kopie erlaubt ist, wird der Programmteil ab Adresse 4118h ausgeführt. Hier wird das Zeigeranpassungsprogramm so geändert, daß dort nun Subtraktionen ausgeführt werden, indem die erwähnten nops, die ja inzwischen Additionsbefehle enthalten, entsprechende Subtraktionsbefehle erhalten.

Jetzt sind alle Überprüfungen durchgeführt worden, und unser Programm kann die eigentliche RAMTOP-Änderung vornehmen. Es muß jedoch an dieser Stelle darauf hingewiesen werden, daß unser Programm eine wichtige Überprüfung unterlassen hat. Wenn RAMTOP erhöht werden soll, müßte es feststellen, ob der entsprechende, höhere Speicherbereich überhaupt vorhanden ist. Wegen der geschilderten Mehrdeutigkeit der Adressen nicht vorhandener Speicherzellen ist eine derartige Überprüfung jedoch unmöglich, wenn man andere Speicherbereiche nicht zerstören will. Wir müssen also selbst darauf achten, daß RAMTOP nur in vorhandene Bereiche gelegt wird.

Betrachten wir nun die drei letzten Teile unseres Programms. Zunächst wird also das Kopieren des GOSUB- und Maschinenstapels durchgeführt. Hierzu wird deren Länge, also die Anzahl der zu kopierenden Zeichen, berechnet. Dann werden Quell- und Zieladresse festgelegt und es wird der *lddr*-Befehl ausgeführt.

Im vorletzten Programmteil, dem Zeigeranpassungsprogramm, wird die Differenz zwischen dem neuen und dem alten RAMTOP-Wert berechnet. Diese Differenz wird zum Inhalt der Zeiger addiert bzw. davon subtrahiert. Nun müssen wir nur noch den Bildschirm auf die richtige Größe bringen, den NMI-Generator wieder einschalten und ins Basic zurückkehren.

Beim Mini-Programm muß man mitdenken

Nachdem wir jetzt wissen, was bei einer RAMTOP-Änderung alles zu beachten ist, wollen wir noch die Mini-Version des Programms betrachten, die in unsere Maschinenprogramme eingebaut, diese nach dem Laden automatisch schützt. Hierbei können wir uns auf das Verringern von RAMTOP beschränken.

Bild 3 zeigt das kleine Programm, das praktisch nur aus den zwei Programmteilen „Kopie durchführen" und „Zeiger anpassen" besteht. Es kann ebenfalls ohne irgendeine Änderung in jedem Speicherbereich betrieben werden. Allerdings muß der jeweils gewünschte RAMTOP-Wert in die Zeilen 4094h und 409Eh eingesetzt werden. In Bild 3 sind z. B. die letzten 3 KByte des 16-KByte-Speichers reserviert. Wollen wir nur 1 KByte reservieren, so müßten die Hex-Codes der genannten Zeilen folgendermaßen lauten: 11007C. Michael Naujoks

ZX-81-Softwaretip:

Soviele Bytes sind frei

Wer sich beim Programmieren eines ZX 81 oft die Frage stellt, wieviel Speicherplatz noch vorhanden ist, der kann sich mit einem Maschinenprogramm (*Bild*) den freien Speicherplatz zwischen STKEND und Maschinenstapel (siehe Sinclair-Handbuch Seite 171) anzeigen lassen. Nach Eingabe der Zeile

1 REM 0000000000000 (13 Nullen)

können die Codes in der Form POKE 16514,33 bis POKE 16526,201 eingetippt werden. Nach LIST wird Zeile 1 dann so aussehen:

1 REM 5TAB COPY T GOSUB ?0RND GOSUB ???TAN

Mit Hilfe von z. B. 9000 PRINT USR 16514 liefert RUN oder GOTO 9000 den gesuchten Wert in Bytes.

Dieser Wert liegt um 100 Byte unter dem größtmöglichen Wert: 37 Bytes erfordern nämlich schon die Zeilen 1 und 9000; der restliche Betrag von L = 63 Byte wird vom Maschinenprogramm abgezogen. Der Wert für L ist in gewissen Grenzen willkürlich gewählt und berücksichtigt, daß beide Speicher-Grenzwerte während der Ausführung eines Befehls oder bei einer Eingabe „atmen". L liefert also etwas Spielraum, der praktisch aber kaum zu nutzen ist.

Ein ZX 81 mit 1 KByte RAM liefert, wenn er nur mit den beiden Zeilen programmiert ist, die Anzeige 755, d. h. der größtmögliche freie Speicherplatz beträgt 855 Byte. Da ein 1 KByte RAM 1024 Byte enthält, fehlen also 169 Byte. Sie sind gleich nach dem Einschalten für die Systemvariablen (125), für den leeren (!) Bildschirm (24) sowie für Maschinenstapel, Marken usw. (20) „verbraucht" worden. Jedes auf dem Bildschirm sichtbare Zeichen verringert den freien Speicher um ein weiteres Byte.

Wird zwischen die Zeilen 1 und 9000 ein Programm eingefügt, erhält man mit RUN den freien Speicherplatz nach Abzug der momentanen Bildspeicherausdehnung und mit GOTO 9000 ohne diesen Abzug. Beim ZX 81 mit 16 KByte RAM ergibt sich dagegen folgende Bilanz:

Von den verfügbaren 16 384 Byte sind abzuziehen: Für die Systemvariablen 125, für den Bildschirm 792, für Maschinenstapel, Marken 20.

Mit dem Programm erhält man dann die Anzeige 15347 (Memotech- oder Sinclair-RAM). Dieser Speicherplatz steht im Gegensatz zur 1-KByte-Version voll für Programme zur Verfügung, da der Bildspeicher vorab reserviert worden ist. Demnach wird mit dem Anstecken des 16 KByte RAM der tatsächliche Programmspeicher um viel mehr als das 16fache vergrößert. Fritz Waschk

Adresse	Code	Mnemonic
16514	33	LD HL,65474
16515	194	
16516	255	
16517	57	ADD HL,SP
16518	237	LD DE,(16412)
16519	91	
16520	28	
16521	64	
16522	237	SBC HL,DE
16523	82	
16524	68	LD B,H
16525	77	LD C,L
16526	201	RET

Maschinenlisting: Dieses kurze Programm ermittelt den freien Speicherplatz zwischen STKEND und Maschinenstapel

Auf Profis programmiert:

Mit mc lernen Sie Computer von Grund auf verstehen. Ausführliche Funktionsbeschreibungen von Rechner-Hardware und gut kommentierte Programm-Listings bieten Ihnen den richtigen Einstieg ins ernsthafte Computern.

Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.

Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.

In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.

Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.
Für DM 7,– bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.
mc können Sie aber auch auf andere Art kennenlernen.
Kostenlos und unverbindlich.
Die Abrufkarte dafür finden Sie an der Umschlagklappe.

mc
Die Mikrocomputer-Zeitschrift

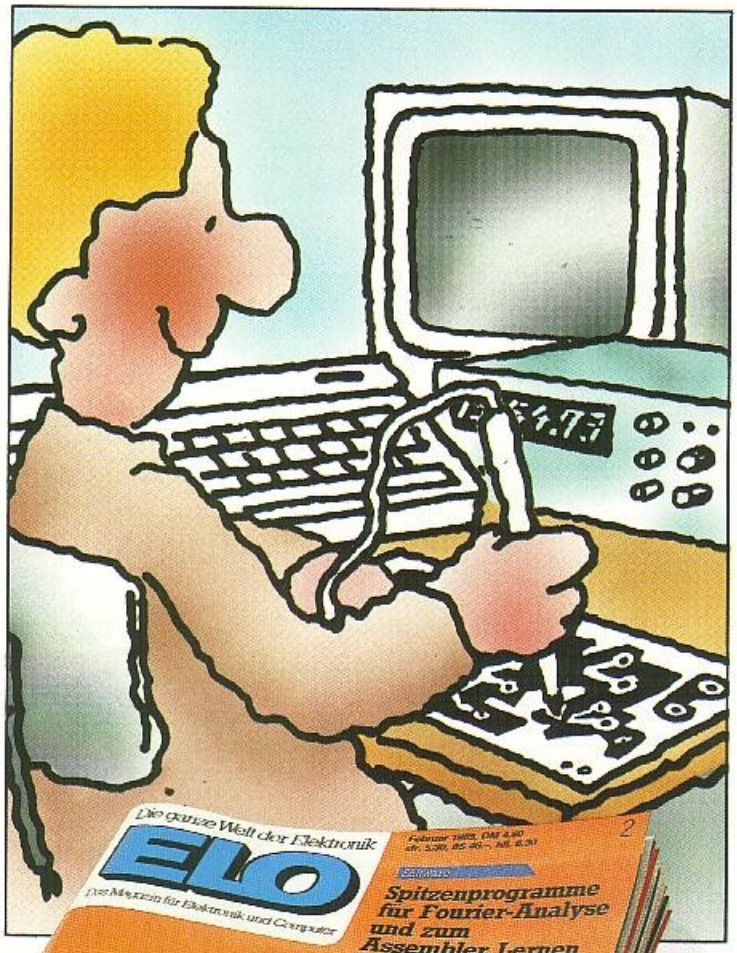
Elektronik verstehen, Computer verstehen – mit der ELO fängt der Spaß an moderner Technik erst so richtig an.

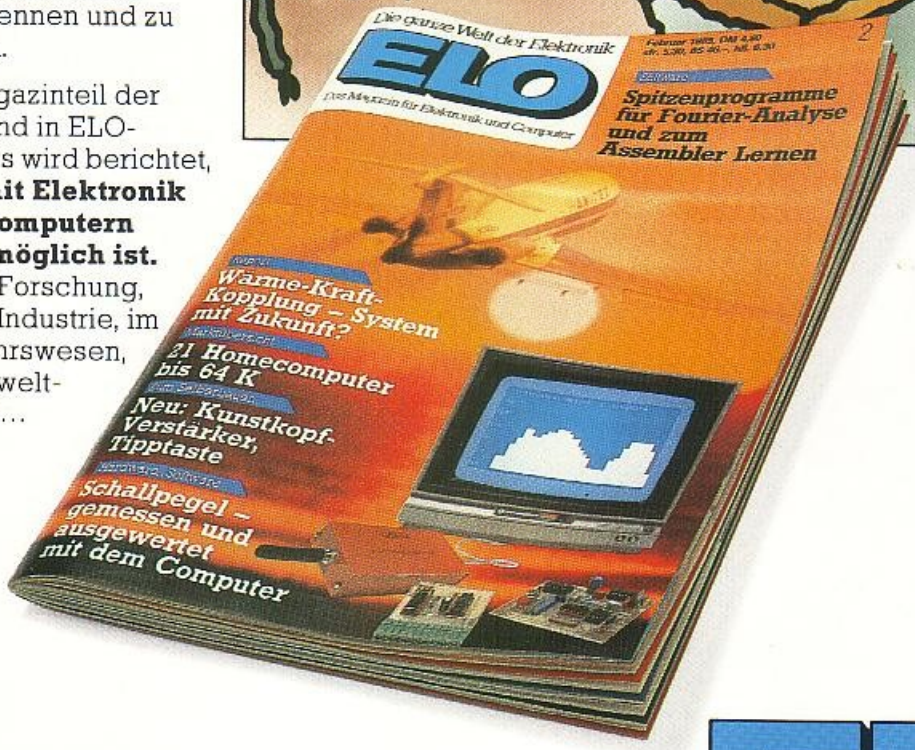
Weil Sie in der ELO leicht zu realisierende **Bauanleitungen** finden, die Ihnen Elektronik praktisch nahe bringen und oft **neue Einsatzbereiche für den Computer** erschließen: Messen, Steuern, Regeln, Erfassen, Auswerten.

Und weil ELO Ihnen helfen wird, Hardware in Ihrem Sinne zu verändern – auch als Anfänger.

Daneben finden Sie leicht verständliche **Beiträge zu Grundlagen der Elektronik,** die sehr schnell dazu beitragen, technische Zusammenhänge zu erkennen und zu nutzen.

Im Magazinteil der ELO und in ELO-Reports wird berichtet, **was mit Elektronik und Computern alles möglich ist.** In der Forschung, in der Industrie, im Verkehrswesen, im Umweltschutz...





Die Rubrik ELO-Service bringt regelmäßig **Marktübersichten,** die Ihnen wichtige Orientierungshilfe im immer unüberschaubarer werdenden Markt elektronischer Produkte sein werden.

Und natürlich finden Sie **ELO-Testberichte:** Über Schach- und Homecomputer, über Video und HiFi-Geräte, über Meßgeräte und alles, was damit im Zusammenhang steht.

Die ELO verbindet die Elektronik mit dem Computer. Wer endlich richtig einsteigen möchte und den Spaß an moderner Technik neu entdecken will, für den gibt's die ELO für DM 5,50 an allen größeren Zeitschriften-Verkaufsstellen.

Noch besser: Sie machen von unserem vorteilhaften Kennenlern-Angebot Gebrauch.

Eine Kennenlern-Karte finden Sie an der Umschlagklappe.

ELO

Das Magazin für Elektronik und Computer